

Pensiero Computazionale e Coding

Bootcamp - Alternanza Scuola Lavoro

[modulo-01]

PENSIERO COMPUTAZIONALE, CODING
E LA SCUOLA

AGENDA

- Coding, Pensiero Computazionale e Apprendimento
 - la scuola costruzionista di Papert
- Coding e Pensiero Computazionale: Quadro elementi principali
 - ruolo del “linguaggio di programmazione”
 - ruolo delle metafore, della narrazione

AGENDA

- **Coding, Pensiero Computazionale e Apprendimento**
 - **la scuola costruzionista di Papert**
- Coding e Pensiero Computazionale: Quadro elementi principali
 - ruolo del “linguaggio di programmazione”
 - ruolo delle metafore, della narrazione

CONTESTUALIZZAZIONI

- Coding nel contesto del Pensiero Computazionale
 - informatica, programmazione, pensiero computazionale
- Pensiero Computazionale nel contesto della scuola
 - informatica, computer e apprendimento

INFORMATICA E COMPUTER

- Informatica
 - scienza che studia i procedimenti effettivi di elaborazione (e trasmissione, memorizzazione) automatica dell'informazione
- Informatica non è (solo) “la scienza dei computer”
 - “informatica sta al computer come l’astronomia sta al telescopio”... o la biologia sta al microscopio (Dijkstra)

PROGRAMMAZIONE E CODING

- Programmazione (informatica)
 - processo che parta da un *problema computazionale* (in senso ampio) per arrivare ad un **programma** effettivamente eseguibile da un computer
- Un informatico deve:
 - analizzare problema
 - costruire un **algoritmo** (sequenza di istruzioni per la soluzione del problema)
 - verificare che l'algoritmo sia corretto, rispettoso dei requisiti, efficiente
 - implementarlo, cioè tradurlo in uno specifico *linguaggio di programmazione* (**coding**)

PENSIERO COMPUTAZIONALE

- *“Modo di pensare volto a risolvere problemi, progettare sistemi, comprendere il comportamento umano basandosi su concetti fondamentali dell’Informatica”*
 - Janette Wing (2006) - Dipartimento di Informatica presso CMU)
 - “Computational thinking” (Papert, 1980)
- Generalizzando
 - *processo mentale di formulazione di problemi e delle loro soluzioni in una forma che sia effettivamente eseguibile da un agente che processa l’informazione*
 - **"quarta abilità di base"**

PENSIERO COMPUTAZIONALE

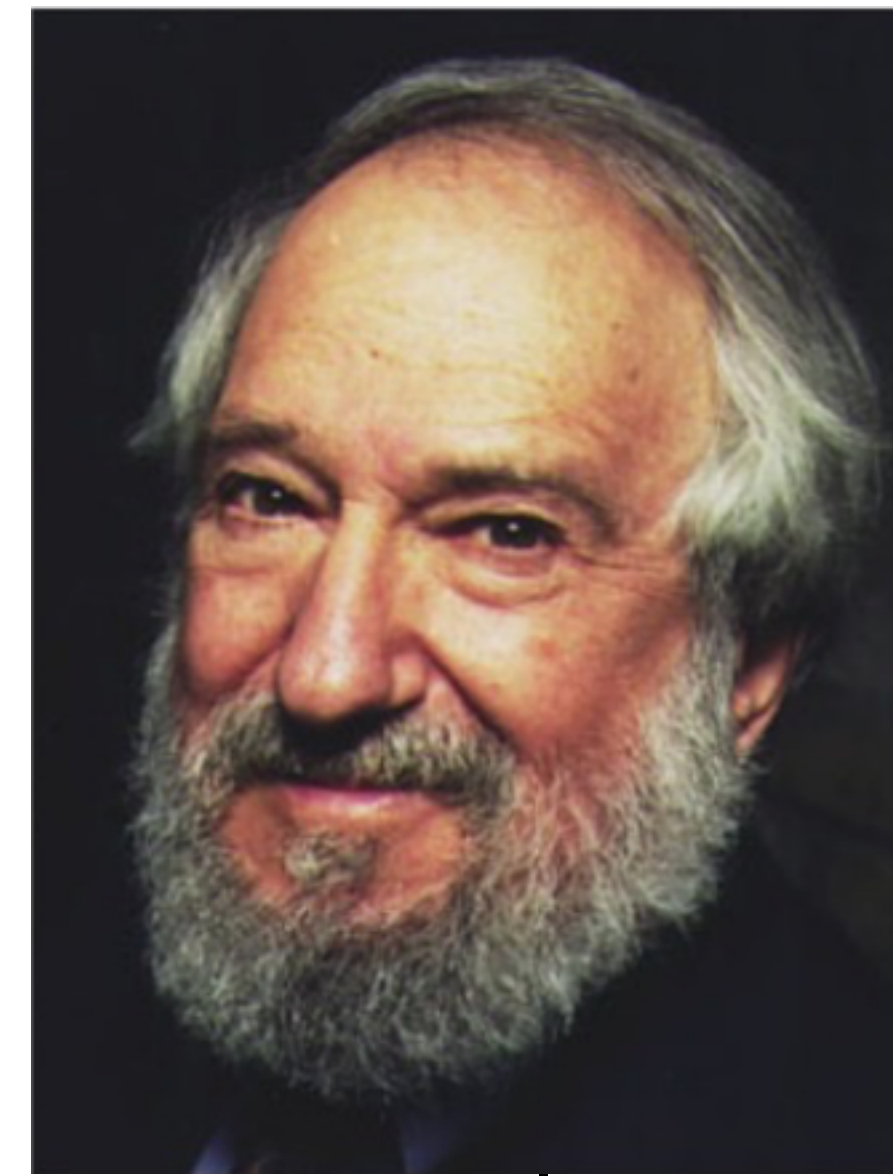
- analizzare i dati, scomporre il problema in problemi più semplici,
- astrarre liberandosi di dettagli inutili, generalizzare per risolvere classi di problemi,
- scegliere modo migliore per rappresentare i dati, immaginare algoritmi efficienti secondo metriche diverse
- scrivere il codice in modo che sia sintetico, leggibile, modulare, manutenibile
- testare e raffinare il codice scritto

INFORMATICA, COMPUTER E *APPRENDIMENTO*

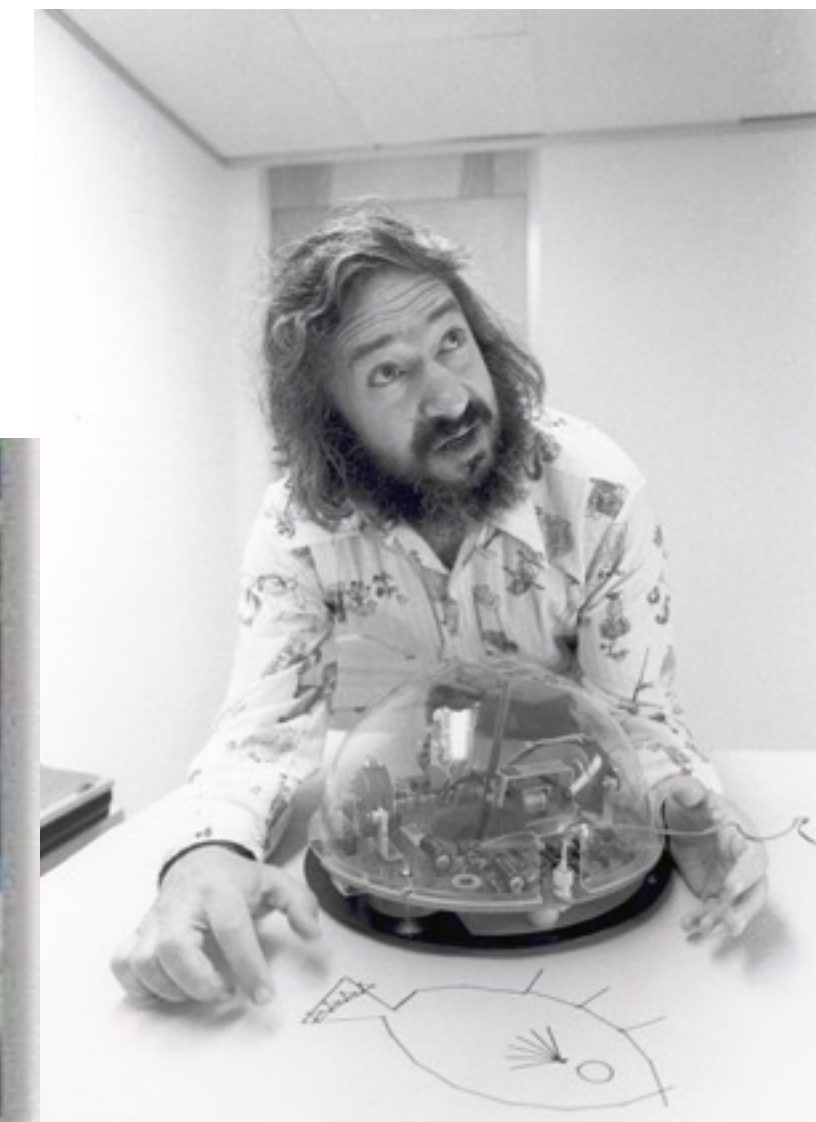
- Computer come tecnologia per l'apprendimento (Papert)
- Pensiero computazionale come meta-competenza

SEYMOUR PAPERT

(1928-2016)



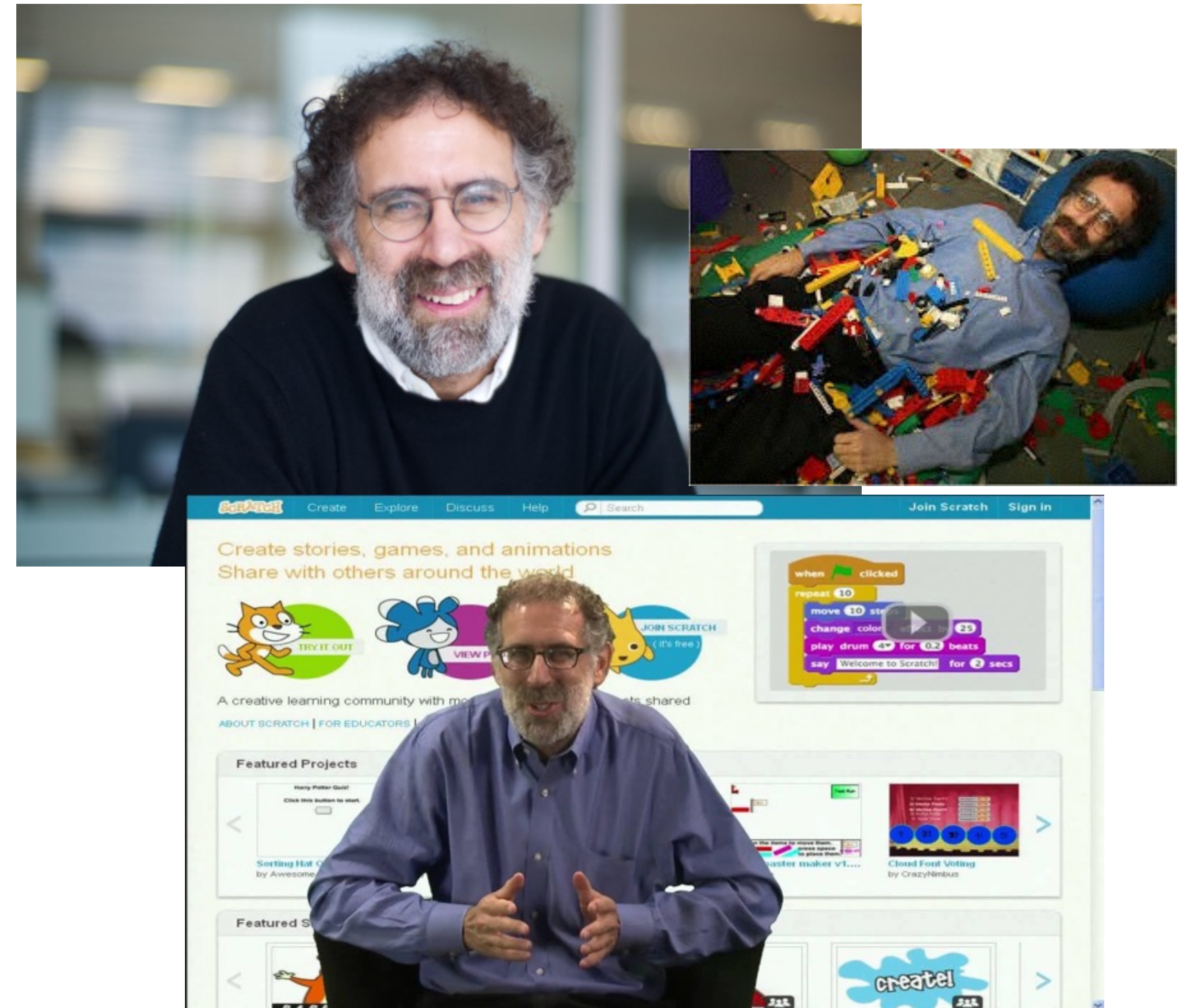
- Matematico, filosofo, pedagogista e informatico
- Ha lavorato prima in Europa con Piaget, poi è divenuto professore al MIT (Massachusetts Institute of Technology)
 - influenzato anche dai lavori della Montessori
- Padre del costruzionismo
- Inventore del linguaggio LOGO



EREDITA' DI PAPERT: MIT Lifelong Kindergarten Group

- **Mitch Resnick**

- ▶ Technologies for Lifelong Kindergarten (1996)
- ▶ Programmable Bricks - Toys to think with (1996)
- ▶ Linguaggio/piattaforma Scratch (2009)



VISIONE COMPLESSIVA

- Visione **costruttivista** della scuola
 - luogo di costruzione di e non di trasmissione della conoscenza
- Ruolo dei computer
 - come “creta” con cui costruire una scultura, è materiale per plasmare, costruire (Paper, 1997)
- Ruolo dell'informatica
 - consente di mettere in luce, grazie al valore cognitivo della progettazione del software(dei programmi) la dimensione creativa dell'uso del computer (Capponi, 2008)

APPROCCIO COSTRUTTIVISTA

- Conoscenza come prodotto costruzione attiva del soggetto
- Ha carattere situato, ancorato al contesto concreto
- Si svolge attraverso particolari forme di collaborazione e negoziazione sociale
- Non è trasmessa, ma **costruita** da chi apprende nella propria mente, dall'interazione fra esperienza che sta vivendo e esperienze precedenti.

RUOLO DEL COMPUTER

- Non è una macchina di informazioni o per gestire informazioni
 - è una macchina per *eseguire progetti, per costruire* (Paper, 1997)
- Rende possibile compiere azioni didattiche prelude agli strumenti tradizionali
 - es: esplorare strutture atomo, simulare effetti clima,...
- Vera essenza: **potere di simulare** (Capponi, Parisi)
 - rendere concreto e personale il formale (Papert)

“IL COMPUTER COME PENNELLO”



- *“Computers will not live up to their potential until we start to think of them **less like television** and **more like paintbrushes**... in my research group at MIT we develop new technologies that follow the tradition of paintbrushes, wooden blocks, and coloured beads, **expanding the range of what children can create, design, and learn**”*

Mitchel Resnick. Computer as Paintbrush: Technology, Play, and the Creative Society. Singer, D., Golikoff, R., and Hirsh-Pasek, K. (eds.), Play = Learning: How play motivates and enhances children's cognitive and social-emotional growth. Oxford University Press. 2006

DAL CALCOLO ALLA SIMULAZIONE

- Passaggio da una **cultura del calcolo** a una **cultura della modellazione e simulazione** (Turkle, 1996)
- Passaggio da una visione dell'informatica come materia da insegnare e computer come strumento da insegnare a una visione in cui l'informatica è *comunicazione, linguaggio* di moderazione di sistemi e situazioni complesse
 - con strumenti che "matematici" / logici in un'accezione ampia, legati alla capacità di base di ragionare e risolvere problemi

COMPUTER E APPRENDIMENTO

- Non solo puro mezzo tecnologico, ma impatto sul metodo con cui può avvenire l'apprendimento (Cavani)
 - permette di abbattere la paura nella nostra cultura rispetto all'apprendimento in generale (Paper)
- Strumento per abbattere muro fra cultura umanistica e scientifica e costruire un *ambiente personale*
 - *un bambino può programmare un computer in un linguaggio che favorisce la **riflessione sul pensiero** e che stabilisce un'intima **relazione con le idee fondamentali** della scienza matematica e l'arte nella costruzione dei modelli intellettuali* (Reggini)

COMPUTER SUL BANCO

- ▶ computer in classe, non nelle aule informatica

“PRINCIPIO DI PAPERT” E BRICOLAGE

- “Principio di Papert” (come battezzato da Minsky, collega MIT)
 - alcuni fra gli stadi più cruciali dello sviluppo mentale sono basati non sulla semplice acquisizione di nuove abilità, ma acquisizione di nuovi *metodi amministrativi* per usare ciò che già si conosce
- Utilità del **bricolage** intesa come metodologia per attività intellettuale e stile di apprendimento (Papert)
 - fonte di idee e modelli per migliorare la capacità di creare, aggiustare le costruzioni mentali
 - usare le cose di cui già si dispone, improvvisare, saper adattare
 - avvicinarsi gradualmente alla meta, procedere per gradi
 - ***indeterminatezza gestita***

RUOLO DELL'ERRORE

- Apprendere dai propri errori
- Correzione dell'errore fa parte del processo di comprensione del programma
 - antitetico al modello di apprendimento *o si è capito o non si è capito*

“FILOSOFIA DEL *DEBUGGING*”

- Gli errori ci aiutano perché ci guidano a studiare ciò che è accaduto, a capire cosa non andava e a sistemare le cose
- Aggiustamento progressivo in cui allievo constata i propri progressi
- Impatto a livello metodologico
 - studenti hanno spesso difficoltà nella risoluzione di un problema perché vogliono affrontarlo *tutto in una volta*
 - induzione di principi di organizzazione **modulare** *del pensiero, del progetto, dei programmi*
 - la scomposizione del problema in parti più semplici facilita notevolmente l'identificazione e rimozione di errori

ERRORI E COLLABORAZIONE

- Gli errori promuovono atteggiamenti collaborativi (Swartz)
 - sia fra bimbi, sia con insegnante
- Con ambienti come Scratch/Logo i bimbi vedono che anche l'insegnante sbaglia e apprende dai propri errori
 - "vera collaborazione", non ingannevole (Papert)
 - condividere il problema e l'esperienza della sua soluzione permette al bambino di imparare dall'adulto non *facendo quello che il maestro dice*, ma *quello che il maestro fa*
- Ambiente scolastico costruttivo è un ambiente in cui l'insegnante apprende insieme al bambino

COSTRUZIONISMO DI PAPERT

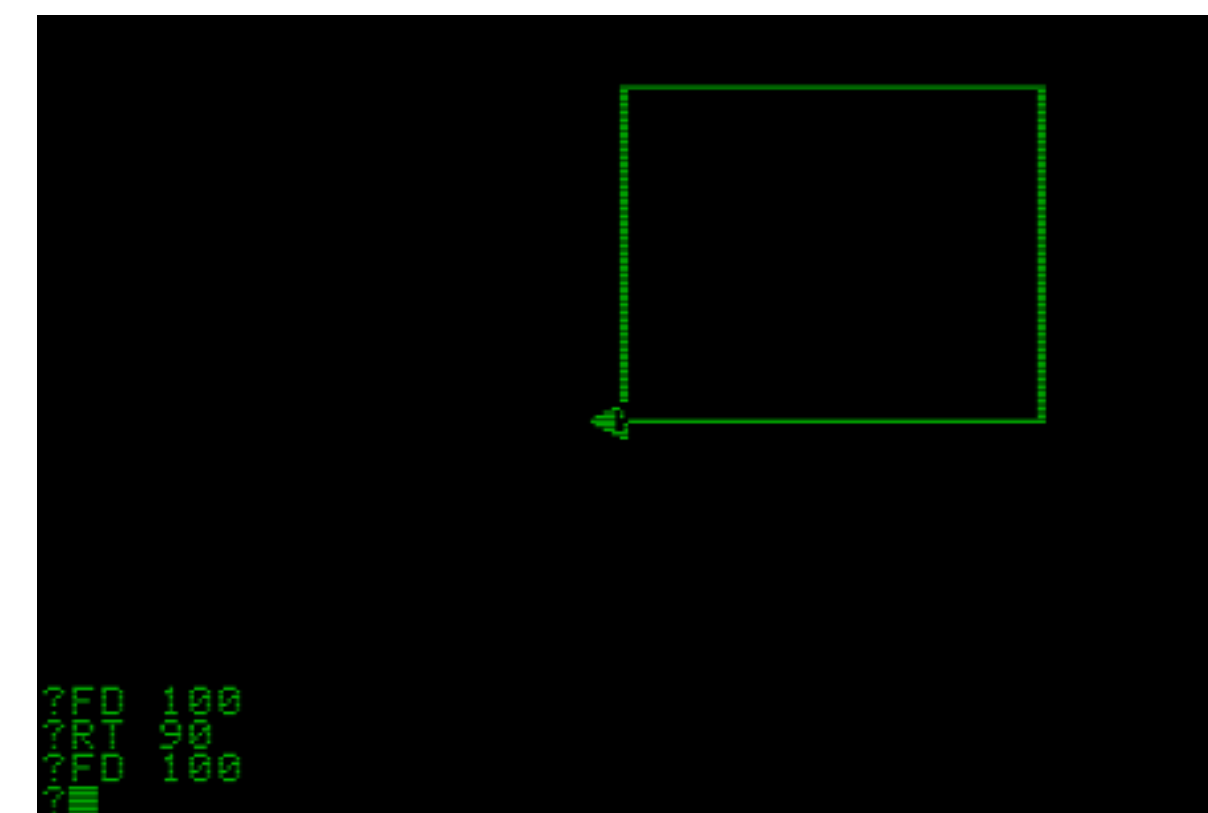
- Estensione dell'**attivismo** di Dewey (learning by doing)
 - continuità progettazione e attualizzazione MA pensando e discutendo attorno a quello che si fa
- Estensione del **costruttivismo** di Piaget (learning by making)
 - costruttivismo non solo mentale, parallela costruzione reale
 - importanza degli oggetti e dispositivi ("**artefatti cognitivi**") che facilitano lo sviluppo di specifici apprendimenti
 - *materiali concreti* affinché la conoscenza acquisita sia tanto più vicina alla realtà

COSTRUZIONISMO: PILASTRI

- Lo studente come protagonista
- Usare per imparare
- Rivalutazione pensiero operatorio concreto su quello formale logico-deduttivo
- Introduzione al concetto di micromondo
- Epistemologia dell'indeterminatezza gestita
 - vs: epistemologia di precisione
- Apologia dell'errore

LINGUAGGIO LOGO

- Linguaggio **Logo** come linguaggio a supporto del costruzionismo (Wally Feurzeig, Papert - ~1980)
 - programma = elenco di istruzioni con le quali far muovere una tartaruga su un piano, con la possibilità di eseguire azioni come tracciare linee, emettere suoni, stampare messaggi..
- Strumento che consente ai bambini (scuole elementari, medie) di utilizzare il computer per ottenere *rapidamente risultati concreti*, ma utilizzando principi matematici e logici rigorosi
 - disegni, musica, poesie...
- Il bambino avverte di avere *il controllo del computer*



DAL LOGO A SCRATCH (E SNAP!)



- Piattaforma Scratch (<https://scratch.mit.edu/>)
 - eredita filosofia e principi del logo, in chiave moderna
 - basata su programmazione visuale
 - piattaforma più utilizzata per coding nelle scuole
- Piattaforma Snap! (<https://snap.berkeley.edu/>)
 - estensione di Scratch, più configurabile e flessibile



MICROMONDI

- Riproduzione effettuata su computer del comportamento di un sistema reale
 - una simulazione digitale basata sullo studio di un caso reale
 - piccoli universi, realtà limitate, luoghi sicuri da esplorare
- Più in generale (Progetto Cogito):
 - *un ambiente/mondo digitale progettato per favorire l'esplorazione e apprendimento (co-costruito) di concetti e competenze multi- e inter-disciplinari*
 - ambiente che crei condizioni analoghe all'educazione pre-scolare

GEOMETRIA DELLA TARTARUGA (Paper, 1980)

- OVVERO: PRINCIPI UTILI NEL DESIGN DI MICROMONDI -

- Principio di ***continuità***
 - primo passo per comprendere un concetto è la possibilità di integrarlo con conoscenze precedenti
- Principio di ***potenza***
 - ambiente di apprendimento deve consentire a chi apprende di concepire progetti carichi di significato, che non avrebbe mai pensato prima
- Principio di ***risonanza culturale***
 - ciò che viene appreso deve avere senso all'interno dell'ambiente sociale nel quale ci troviamo

“LEARN TO CODE, CODE TO LEARN”

(Resnick, EdSurge 2013)



- Coding (programmazione) come estensione della capacità di scrivere e creare
 - *si insegna ai bimbi a scrivere non perché diventino necessariamente scrittori, a fare calcoli non perché diventino matematici.. analogamente non si insegna il coding (la programmazione) perché diventino informatici*
 - il coding permette di scrivere nuovi tipi di cose - storie interattive, giochi, animazioni, simulazioni...

CIRCOLO VIRTUOSO (PROGETTO COGITO)

SOGGETTO

MEZZO

OGGETTO/OBIETTIVO

Studenti
(e docenti)

Pensiero
computazionale
e coding

metacompetenza

Competenze in
ambiti disciplinari
eterogenei

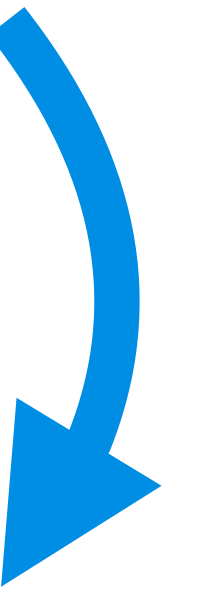
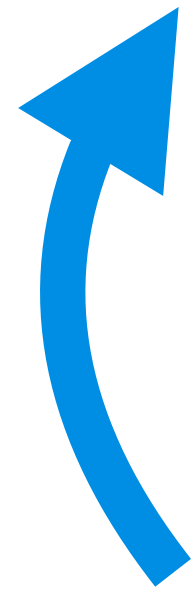
Studenti
(e docenti)

Pensiero
computazionale
e coding

metacompetenza

Competenze in
informatica

LEARN TO CODE CODE TO LEARN



AGENDA

- Coding, Pensiero Computazionale e Apprendimento
 - la scuola costruzionista di Papert
- **Coding e Pensiero Computazionale: Quadro elementi principali**
 - **ruolo del “linguaggio di programmazione”**
 - **ruolo delle metafore, della narrazione**

PENSIERO COMPUTAZIONALE

- **Concetti**
- **Metodi**
- **Pratiche**

PENSIERO COMPUTAZIONALE

QUADRO DEI **CONCETTI**

- Sequenza
- Condizionali
- Ripetizioni
- Eventi
- Parallelismo
- Operatori
- Espressioni matematiche e logiche
- Variabili
- Dati (collezione, analisi, rappresentazione)

PENSIERO COMPUTAZIONALE

QUADRO DEI **METODI**

- Decomposizione
- Astrazione
- Riconoscimento pattern e generalizzazione

PENSIERO COMPUTAZIONALE

QUADRO DELLE **PRATICHE**

- Essere incrementali e iterativi
- Testing e debugging
- Riuso e mixing
- Attenzione all'efficienza, calcolabilità e complessità

PENSIERO COMPUTAZIONALE

QUADRO DELLE **NUOVE PROSPETTIVE (PUNTI DI VISTA)**

- Esprimere se stessi
- Connettersi
- Farsi domande
- Saper gestire la complessità e i problemi difficili
- Tollerare l'ambiguità e i problemi aperti

CODING

- Ideazione e scrittura di un **programma**
 - programma = sequenza di istruzioni per raggiungere un certo obiettivo, svolgere un certo compito
- I programmi *implementano* **algoritmi**
 - algoritmo = insieme ordinato di passi o strategia per risolvere operazionalmente un problema o per costruire un sistema
- I programmi sono basati su un **linguaggio di programmazione**
 - usato per descrivere, implementare gli algoritmi

LINGUAGGI DI PROGRAMMAZIONE

- “Linguaggi *formali*”
 - caratterizzati da regole esatte rigorose che permettono di scrivere il programma in maniera *non ambigua*
- Due macro-aspetti
 - **sintassi** - regole di come le istruzioni devono essere scritte, vocabolario
 - **semantica** - definisce cosa fa il programma, quando mandato in esecuzione da un certo esecutore (es: computer)

LINGUAGGI INTERPRETATI E COMPILATI

- Due macro suddivisioni
 - linguaggi ***interpretati***
 - esecuzione = interpretazione del programma
 - interprete = programma che legge, interpreta ed esegue il programma
 - linguaggi ***compilati***
 - fase di “compilazione” = traduzione del programma in linguaggio direttamente comprensibile ed eseguibile sul sistema operativo/processore
 - esecuzione senza necessità di interpretazione

LINGUAGGI VISUALI E A BLOCCHI

- Linguaggi **visuali** vs **testuali**
 - equivalenti dal punto di vista espressivo
- Linguaggi visuali **a blocchi** (es: Scratch, Snap!, Blockly)
 - ispirazione dalle costruzioni (Lego)
 - istruzioni come blocchi componibili
 - “affordance” - incastri, colore
 - abbattano alcuni elementi di complessità, astraendo

LINGUAGGIO COME MEZZO DI ESPRESSIONE E COMUNICAZIONE

- Linguaggi di programmazione sono inquadrabili anche come *interfacce fra l'uomo e la macchina/il computer*
- ovvero: come **linguaggi di comunicazione**
 - è possibile “dialogare” con un computer senza conoscere com'è fatto dentro o come funziona

LINGUAGGI E MICROMONDI

- Ogni linguaggio definisce un ***livello di astrazione***
 - insieme dei concetti di prima classe usati per esprimersi
- Possiamo definirci il nostro linguaggio, al giusto livello di astrazione per un certo tipo di scopo, di dominio
 - in letteratura informatica si parla di *Domain Specific Language*
- Collegamento con i micromondi
 - **ogni micromondo implicitamente o esplicitamente definisce un linguaggio**

MICROMONDI E LINGUAGGI

- definisce il livello di astrazione
 - insieme dei concetti con cui rappresentiamo il micromondo e le sue dinamiche
- può essere costituito da *simboli* di tipo diverso
 - parole, figure, ...
- ruolo chiave nel governare la “complessità” del micromondo

ESEMPI

- Esempi dal progetto COGITO
 - micromondo tartarughe, insalate e buche
 - micromondo disegno
 - micromondo matematica
 - micromondo parole

METAFORE E CHIAVI NARRATIVE

- Introduzione di metafore e “narrazioni” utili a guidare l’ideazione e progettazioni di programmi
 - “invarianti” rispetto allo specifico linguaggio usato
- Esempi
 - **Personificazione**
 - **“Teatralizzazione”**

PERSONIFICAZIONE

- Identificazione di un oggetto, un *personaggio* con cui pensare e interpretare il micromondo, con cui *identificarsi*, “mettersi nei suoi panni” per capire cosa vuol dire eseguire i comandi
 - tartaruga
- Programmare = comunicazione con l’oggetto identificato
 - insegnare alla tartaruga come risolvere un problema o come svolgere un certo compito, in modo *operazionale*

TEATRALIZZAZIONE

- Micromondo come spettacolo teatrale da mettere in scena
 - **personaggi, palcoscenico**
- Il programmatore/i programmatori sono i **registi** dello spettacolo
 - coding => scrivere il copione (script) dei personaggi
 - sceneggiatura = programma del micromondo nel suo complesso
- Testing e debugging => fare le prove, studiare lo spettacolo vedendolo svolgersi, ragionare su come migliorarlo

TEATRALIZZAZIONE

- Elementi del programma: **attori, scena**
 - In Scratch/Snap!: attori => “sprite”, scena => “stage”

TEATRALIZZAZIONE

- Coding => scrivere il **copione (script)** degli attori e della scena
 - copione = programma, implementa un algoritmo
 - tipicamente associato ad uno o più scopi o compiti (obiettivi dell'attore)
- Linguaggio con cui scrivo lo script => linguaggio per comunicare (con gli attori) e modellare (rappresentare lo spettacolo)

TEATRALIZZAZIONE

- Visione del singolo attore vs. visione d'insieme, di *sistema*
 - “sceneggiatura”
 - programmare la visione d'insieme del micromondo nel suo complesso
 - “scenografia”
 - design della scena dove si muovono gli attori

RIFERIMENTI BIBLIOGRAFICI SELEZIONATI

- Massimo Capponi. *Un giocattolo per la mente. L'informatica cognitiva di Seymour Papert*. Morlacchi editore, 2008
- Rita Marchignoli e Michael Lodi. EAS e Pensiero Computazionale. Fare coding alla scuola primaria. ELS Scuola, 2016
- Seymour Papert. *Mindstorms. Bambini, computers e creatività*, 1980
- Seymour Papert. *Connected family. Come aiutare genitori e bambini a comprendersi nell'era di Internet*, 2003