ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

# PENSIERO COMPUTAZIONALE E CODING IN UN LICEO CLASSICO: SPERIMENTAZIONE DI MICROMONDI IN SNAP! E COSPACES EDU E PROPOSTA DI UNA NUOVA PIATTAFORMA INTEGRATA

*Elaborato in*
SISTEMI EMBEDDED E INTERNET OF THINGS

*Relatore*
Prof. ALESSANDRO RICCI

*Presentata da*
ANGELA SPERANZA

*Correlatori*
Prof. ELVIS MAZZONI
Dott.ssa YLENIA BATTISTINI

Anno Accademico 2022 – 2023

*A mia sorella Alessandra,*
*mia guida*
*e stella polare nella notte*

# Indice

# Introduzione

*"**Ludus** iuvat etiam quia signum intelligentiae est: ludo discipuli libenter discent et sine molimento cognoscent rerum naturam."*

"Il **gioco** giova, anche perché è segno di intelligenza: con il gioco gli alunni impareranno volentieri e conosceranno senza grande sforzo la natura delle cose."

La citazione appena riportata proviene dall'"*Istitutio Oratoria*" di Quintiliano, un autore latino che visse e pubblicò i suoi scritti intorno alla metà del primo secolo dopo Cristo. Prima di passare alla stesura di opere, Quintiliano fu un insegnante di retorica che lavorava a stretto contatto con i suoi studenti. Aveva molto a cuore il mondo dell'educazione dei giovani, tanto che nell'"*Istitutio Oratoria*" non si limitò ad elencare le caratteristiche della retorica e a creare l'immagine del perfetto oratore, ma descrisse dettagliatamente anche i compiti del precetto e i migliori metodi d'insegnamento, approfondendo anche questioni pedagogiche per seguire l'alunno fin dalla tenera età.

In particolare, Quintiliano riconosceva nel **gioco** un importante valore formativo: attraverso le attività ludiche, infatti, si poteva migliorare l'eloquenza e la capacità di persuasione ma anche la gestione delle emozioni e dei rapporti interpersonali. Quintiliano è solo uno dei tanti esempi di autori antichi che hanno trattato questa tipologia di temi. Nell'antica Roma, così come nell'antica Grecia, il concetto di gioco come strumento educativo era già ben radicato nella società.

Infatti, era comune pensare che il "***ludus***" andasse oltre l'intrattenimento infantile e che in realtà rappresentasse una fonte da cui attingere per ricavare importanti lezioni di vita, nonché per sviluppare competenze utili per la crescita e per la serena partecipazione alla vita sociale. Da sempre, quindi, il gioco non è separato dall'educazione, ma piuttosto costituisce un elemento fondamentale del percorso scolastico: tramite un approccio ludico si possono, quindi, plasmare le menti degli studenti, per farli crescere come individui con-

sapevoli di loro stessi e degli altri, parlando una lingua adatta a loro e che permette di conciliare studio e divertimento.

Non a caso, infatti, il termine "***ludus***" sul dizionario Latino-Italiano "IL", riporta, tra i diversi significati - oltre a "gioco" - anche "**scuola**".

Ma dove si colloca il *gioco* nella scuola odierna? La risposta potrebbe celarsi dietro una visione innovativa della didattica, che vede nell'uso delle nuove tecnologie un'opportunità per rendere l'apprendimento più coinvolgente e stimolante.

Negli ultimi anni, il mondo dell'educazione ha subito una profonda trasformazione, dovuta principalmente all'avvento delle nuove tecnologie. Queste ultime hanno rivoluzionato il modo di apprendere e insegnare, rendendo possibile l'accesso a un'infinità di risorse e strumenti che permettono di personalizzare il percorso formativo e di rendere l'apprendimento più coinvolgente e stimolante.

In particolare, la **programmazione** e il **pensiero computazionale** sono diventati argomenti di grande interesse nel mondo dell'educazione, in quanto permettono di sviluppare competenze trasversali e di promuovere l'acquisizione di conoscenze in un contesto più leggero rispetto a quello tradizionale. In questo contesto, i **micromondi digitali** rappresentano un'ottima soluzione educativa, valida anche in caso di insegnamenti complessi e progetti interdisciplinari.

L'obiettivo primario di questa tesi è verificare, attraverso un apposito percorso sperimentale, come la creazione di micromondi digitali possa costituire un'attività di valore,s sia dal punto di vista educativo che da quello della motivazione e dell'interesse degli studenti. In particolare, si intende contestualizzare questa attività in un ambiente scolastico di indirizzo umanistico, come il **Liceo Classico**.

Il percorso tracciato parte dai concetti di "pensiero computazionale" e "*coding*". Questi strumenti costruiscono le fondamenta dei micromondi, nati grazie al rivoluzionario contributo di Seymour Papert. Queste risorse sono giunte ai giorni nostri, in un contesto in cui le scuole italiane e i Ministeri dell'Istruzione stanno cercando di integrare le nuove tecnologie nei programmi di studio. In questo modo, si esplora l'evoluzione di un approccio pedagogico che affonda le radici nel passato e si adatta alle sfide contemporanee della formazione, avviando il percorso verso la Scuola 4.0.

I protagonisti di questo percorso sono stati gli studenti del Liceo Classico "Vincenzo Monti" di Cesena, nel dettaglio gli alunni della classe 1Bc e 2Bc, e le loro insegnanti - la prof.ssa Piraccini e la prof.ssa Ranieri.

Un fattore interessante di questa sperimentazione è stato proprio il contesto in cui è stata svolta: il Liceo Classico, infatti, è un istituto scolastico che si caratterizza per un'offerta formativa improntata allo studio delle lingue

classiche, della letteratura e della storia antica, e poco incline all'introduzione di competenze informatiche e digitali.

In questo contesto, quindi, la programmazione e il pensiero computazionale integrati all'interno di micromondi immersivi rappresentano ancor di più un'innovazione didattica, la quale può portare a una maggiore interdisciplinarità e a una maggiore consapevolezza delle potenzialità delle nuove tecnologie. Il percorso formativo intrapreso, inoltre, ha permesso di analizzare e mettere in luce importanti aspetti legati all'esperienza d'uso che circonda gli strumenti utilizzati, in particolare **Snap!** e **CoSpaces Edu**.

Questi ultimi, *tools* rappresentativi nel panorama scolastico, hanno permesso la creazione di micromondi bellissimi e coinvolgenti, ma hanno anche mostrato alcune limitazioni e difficoltà di utilizzo, soprattutto per quanto riguarda la collaborazione in tempo reale e la scarsa flessibilità nella gestione di alcune funzionalità. In questo contesto, nasce l'idea di cercare soluzioni alternative per la creazione di micromondi digitali, che permettano di superare le limitazioni riscontrate e di offrire un'esperienza d'uso più completa e soddisfacente.

La presente tesi di laurea si articola in quattro parti principali. Il primo capitolo è dedicato al pensiero computazionale e al *coding* come strumenti educativi: vengono presentati i concetti di base e le motivazioni che hanno portato alla diffusione di queste tematiche nel mondo dell'educazione moderna. Sempre in questo capitolo viene anche presentato il concetto di "micromondo", cercando di mettere in evidenza come questi ambienti di apprendimento siano in grado di favorire lo sviluppo di competenze trasversali e di promuovere l'acquisizione di conoscenze in modo divertente e coinvolgente.

Il secondo capitolo, invece, è dedicato alla descrizione tecnica dei due strumenti utilizzati nella fase sperimentale nella creazione dei micromondi digitali: Snap! e CoSpaces Edu. Si denotano le caratteristiche più importanti di questi due ambienti di programmazione visuale e ne vengono descritte le modalità di utilizzo e le funzionalità principali.

Il terzo capitolo è dedicato alla descrizione dell'intera fase sperimentale: si delinea il percorso formativo svolto con gli studenti coinvolti e si mostrano i risultati ottenuti. Preminentemente, si pone particolare attenzione sia ai risultati concreti delle attività svolte, sia alle impressioni e ai feedback raccolti dagli studenti e dagli insegnanti. In questo capitolo si ritrova anche una analisi critica degli strumenti utilizzati, basata soprattutto sulle esperienze dirette e sulle impressioni raccolte durante la fase finale della sperimentazione.

Il quarto capitolo è dedicato alla ricerca di soluzioni alternative per la creazione di micromondi digitali: a valle delle attività sperimentali svolte, si è deciso di approfondire la ricerca di strumenti più adatti e più performanti

rispetto a quelli utilizzati nella fase sperimentale. Si propone, quindi, una soluzione integrata per uno strumento educativo digitale completo, che permetta di creare progetti collaborativi in tempo reale e di interfacciarsi con tecnologie per la realtà virtuale.

Dopo i primi quattro capitoli, si procede con una sezione dedicata alle conclusioni di questo lavoro. Successivamente, vengono espressi i ringraziamenti, rivolti a coloro che hanno contribuito e fornito costante supporto per la realizzazione di questa tesi. Infine, è presente la bibliografia, che include *papers*, articoli e libri fondamentali per il successo di questo lavoro.

# Capitolo 1

# Pensiero computazionale, *Coding* e Micromondi

## 1.1 Educare al Digitale: un "ponte" tra lo sviluppo umano e l'evoluzione tecnologica

In una società come quella odierna, ormai completamente immersa nella tecnologia, è impensabile che anche il mondo dell'istruzione non ne sia fortemente influenzato. Gli stessi studenti, che al giorno d'oggi sono tutti nativi digitali, stanno crescendo accompagnati da dispositivi tecnologici sempre più avanzati e da un'ampia varietà di applicazioni e servizi digitali. In questo contesto, si ritiene di fondamentale importanza che la scuola, il centro di gravità della vita di ogni studente, sia in grado di fornire ai giovani tutti gli strumenti che servono non soltanto a comprendere il mondo digitale che hanno tra le mani, ma anche a saperne fare un uso responsabile nonché fruttuoso per quanto riguarda la propria crescita personale e professionale.

Guardando al futuro, dunque, l'introduzione delle nuove tecnologie digitali come supporti per una didattica innovativa non può più essere considerata opzionale da parte delle scuole e dei docenti, ma piuttosto un necessario investimento per fare in modo che le nuove generazioni siano in grado di sfruttare al massimo le opportunità che la tecnologia può offrire. Inoltre, è fondamentale riconoscere il **valore intrinseco di queste tecnologie**, per calarlo all'interno di **nuovi approcci metodologici** che vanno ben oltre l'utilizzo passivo di questi strumenti.

Infatti, sono tanti gli ambiti di applicazione in cui si possono ritrovare aspetti più profondi trasmessi da questi approcci. Si pensi soltanto alla sicurezza online e alla privacy: gli utenti di Internet, specialmente se giovani e inesperti, devono essere informati sulle minacce digitali e imparare come proteggere se

stessi e i propri dati sensibili. In secondo luogo, si guarda anche alla promozione di un comportamento etico e responsabile online, incoraggiando l'empatia, il rispetto e la tolleranza nelle interazioni sociali. Per quanto riguarda, invece, lo sviluppo di competenze e *soft skills*, è importante che sin dagli anni scolastici venga promosso e incentivato il pensiero critico, il *problem solving* e la creatività, incoraggiando gli individui a utilizzare gli strumenti digitali per esprimere idee, creare contenuti originali e collaborare online. Questa visione educativa non solo promuove lo sviluppo di nuove competenze, ma anche una partecipazione attiva nella società digitale in continua evoluzione.

## 1.2 Pensiero computazionale

Con l'espressione "pensiero computazionale" si fa riferimento ad un concetto che negli ultimi decenni ha guadagnato un'importanza sempre maggiore nell'ambito della Scuola 4.0. Questo termine si compone di due parole chiave. La prima, "pensiero", si riferisce alla caratteristica che rende l'essere umano unico rispetto a tutti gli altri esseri viventi, ovvero la capacità di ragionare, analizzare dati e situazioni e risolvere problemi. La seconda, "computazionale", deriva dall'inglese *to compute*, e può essere tradotta come "calcolare". Questi due termini, insieme, definiscono un concetto che molti, al giorno d'oggi, considerano la **quarta abilità fondamentale per l'apprendimento**, insieme alla lettura, alla scrittura e all'aritmetica. Tuttavia, questa speciale competenza non nasce in concomitanza con l'utilizzo del computer, ma si tratta di un insieme di processi mentali molto più antichi e sottili che non riguardano soltanto il mondo dell'informatica, ma possono essere applicati a qualsiasi contesto della vita quotidiana.

Attualmente, il concetto di pensiero computazionale più diffuso è stato proposto da **Jeannette Wing**, una figura di spicco nel campo dell'informatica educativa. Wing, a sua volta, ha esteso e modernizzato l'approccio originario di **Seymour Papert**, riconosciuto come il padre tanto del termine "pensiero computazionale" quanto del concetto di micromondo. La filosofia costruzionista di Papert rappresenta un'evoluzione delle teorie costruttiviste di **Jean Piaget** e socio-costruttiviste di **Lev Vygotskij**, due eminenti psicologi e pedagogisti del secolo scorso. Le loro teorie, ancor oggi straordinariamente attuali, hanno gettato le fondamenta per facilitare lo sviluppo umano in un'epoca fortemente caratterizzata dalla digitalizzazione. La continuità e l'attualità di questi principi dimostrano la loro duratura rilevanza nella comprensione e nell'indirizzamento della crescita individuale nell'era digitale.

### 1.2.1 Psico-pedagogia dello sviluppo: Piaget e Vygotskij

Jean Piaget fu uno psicologo, biologo, pedagogista e filosofo svizzero. Piaget era convinto che il processo di apprendimento può essere esplicato come la costruzione mentale di rappresentazioni funzionali del sistema con il quale si interagisce; ciò, in realtà, si applica sia al bambino che all'adulto. L'esperienza empirica aiuta l'essere umano a costruire materiali mentali di riferimento da "consultare" in relazione alla realtà esterna: questo, per giunta, avviene in modo particolarmente efficace e soddisfacente quando vi è anche una realizzazione pratica e concreta di questi concetti.

Piaget pensava che **lo sviluppo cognitivo del bambino è diviso in vari stadi**, sequenziali l'uno all'altro. Questo processo lineare non è importante solo per lo sviluppo pensiero del bambino, ma anche per il suo linguaggio.

Secondo Piaget, il linguaggio si sviluppa in conseguenza al pensiero: il bambino, tramite i meccanismi di assimilazione e accomodamento di cui si è parlato poc'anzi, svilupperà il pensiero e, di conseguenza, riuscirà a sviluppare anche il linguaggio.

La capacità rappresentazionale si sviluppa nella prima infanzia: durante lo stadio senso-motorio, il piccolo possiede degli schemi molto semplici che gli fanno percepire il mondo esterno e gli permettono di rispondere agli stimoli percettivi in maniera riflessa.

Dai 18 mesi ai 7 anni si consolidano degli schemi molto più complessi, che sono un'evoluzione degli schemi precedenti e che permetteranno al bambino lo sviluppo di rappresentazioni mentali degli oggetti percepiti e di conseguenza anche lo sviluppo del linguaggio. In questa fase della vita, il linguaggio è prettamente egocentrico: ciò vuol dire che il bambino non ha cognizione che, al di là del suo punto di vista, ne possano esistere altri. Per questo motivo non si preoccupa di adattare il suo linguaggio a quelle che possono essere le esigenze degli interlocutori con il quale si trova a parlare.

Nella seconda infanzia invece, che va dai 7 agli 11 anni, il pensiero rimane sempre agganciato alla concretezza, ma diventa meno egocentrico. Piaget pensava che, in questa fase, difficilmente il bambino riesce ancora a svincolarsi dal suo punto di vista e a guardare gli altri punti di vista: tuttavia, il linguaggio diventa significativamente più socializzato.

Dai 12 anni in su, invece, con lo stadio logico-astratto, anche il pensiero diventerà ipotetico-deduttivo e astratto e il linguaggio sarà certamente più evoluto.

Dunque, secondo lo psicologo svizzero, i piccoli imparano da subito a comunicare, ma solo in un secondo momento riescono a "filtrare" i propri pensieri nei loro discorsi. Tuttavia, come si può facilmente dedurre, Piaget cerca di

forzare il percorso di sviluppo dei bambini in delle "fasi" legate all'età, quasi senza considerare l'ambiente sociale che circonda il bambino. A questa visione così schematica si oppone quella di Lev Vygotskij.

Lev Vygotskij, definito dal filosofo Stephen Toulmin il "Mozart della psicologia", fu uno psicologo e pedagogista russo, che operò principalmente durante il regime sovietico. Il regime mal vedeva il pensiero occidentale, considerato troppo concentrato sullo sviluppo del pensiero nei bambini. Vygotskij, pur aderendo a questa filosofia, non smise mai di confrontarsi queste idee. Durante la sua breve vita, numerosi furono gli studi e le pubblicazioni che fece sulla psicologia, sui bambini e sulla psicologia educativa. La sua opera più importante risale al 1934, l'anno della sua morte: "Pensiero e linguaggio". L'opera, un paio d'anni dopo, nel 1936, fu condannata dal Comitato Centrale del partito comunista sovietico perché considerata troppo a favore di quelle che erano tutte le teorie occidentali sulla psicopedagogia scientifica.

In questo scritto, Vygotskij, però, prende decisamente le distanze da Piaget. Secondo lo studioso, quello che analizzava Piaget era un bambino che scopre il mondo, come "uno scienziato" isolato nel suo laboratorio che sperimenta l'ambiente che lo circonda. Vygotskij critica molto questa visione, in quanto, secondo lui, l'**aspetto della socialità del bambino è quello più importante**.

Infatti, **il bambino cresce nell'interazione con gli altri**, soprattutto nei primi anni della sua vita, quando si interfaccia con gli adulti. Lo sviluppo umano, quindi, si configura per Vygotskij come un prodotto storico-sociale [5] nel quale, appunto, ha un ruolo fondamentale proprio l'adulto. La persona adulta crea una sorta di "impalcatura" sulla quale poi il bambino metterà tanto di suo. Questa impalcatura verrà denominata da Jerome Bruner "*scaffolding*" [4].

Vygotskij sosteneva che si nasce con quattro funzioni mentali elementari: attenzione, sensazione, percezione e memoria. L'ambiente sociale e culturale che ci circonda ci permette di utilizzare queste abilità per sviluppare e acquisire funzioni mentali superiori.

Questo processo avviene idealmente nella **zona di sviluppo prossimale**.

In primo luogo, c'è quello che possiamo fare da soli: si tratta del bagaglio di competenze e conoscenze attuali, ovvero l'**area di sviluppo attuale**. In secondo luogo, c'è la zona di sviluppo prossimale, che rappresenta ciò che il bambino può fare con l'aiuto di un adulto, di un amico, della tecnologia e di quello che Vygotskij chiamava, in generale, l'"altro più consapevole". Infine, c'è ciò che è fuori dalla nostra portata.

Secondo Vygotskij **solo chi impara con l'assistenza di un mentore capace può raggiungere il pieno potenziale delle proprie capacità**.

Vygotskij quindi riteneva che all'interno della zona di sviluppo prossimale l'apprendimento può precedere lo sviluppo, il che significa che ogni bambino è in grado di apprendere abilità che vanno oltre la sua naturale maturità.

Infine, per quanto riguarda il rapporto tra pensiero e linguaggio, Vygotskij rispose alla teoria di Piaget affermando l'esistenza di una connessione esplicita tra il discorso e i concetti mentali. Egli sosteneva che il discorso interno si sviluppa a partire da quello esterno attraverso un processo graduale di interiorizzazione. Ciò significa che il pensiero stesso si sviluppa come il risultato della conversazione, pertanto i bambini più piccoli che non hanno completato questo processo possono solo "pensare ad alta voce". Una volta completato il processo, il linguaggio interiore e quello parlato diventano indipendenti. Vygotskij ha lasciato il seguente consiglio agli educatori: "Dando agli studenti la possibilità di esercitarsi a parlare con gli altri, diamo loro una cornice per pensare da soli".

### 1.2.2 Seymour Papert

*If children really want to learn something, and have the opportunity to learn it in use, they do so even if the teaching is poor. For example, many learn difficult video games with no professional teaching at all! Others use Nintendo's system of telephone hot lines or read magazines on strategies for games to find the kind of advice for video games that they would get from a teacher if this were a school subject.*

*Se i bambini vogliono davvero imparare qualcosa e hanno l'opportunità di imparare facendo, loro lo fanno anche se l'insegnamento è di scarsa qualità. Per esempio, molti imparano videogiochi difficili senza alcun insegnamento professionale! Altri utilizzano il sistema di Nintendo di linee telefoniche o leggono riviste sulle strategie per i videogiochi, il tipo di consigli che avrebbero ottenuto da un insegnante se questa fosse una materia scolastica.*

[1]

Il primo ad avere utilizzato il termine "Pensiero computazionale" fu **Seymour Papert**, matematico, informatico e pedagogista sudafricano naturalizzato statunitense. Papert può essere considerato uno dei padri dell'informatica educativa, ovvero quella branca dell'informatica che si occupa di studiare e sviluppare metodi e strumenti per l'insegnamento e l'apprendimento in un nuovo contesto digitale.

[1] Seymour Papert, *The Children's Machine: Rethinking School in the Age of the Computer*, 1993. [16]

Papert fu allievo di Jean Piaget. Piaget, di cui si è parlato nel paragrafo precedente, viene anche ricordato per il suo contributo nel campo dell'educazione e, in particolare, per le sue teorie costruttiviste, fortemente contrapposte ad un'altra tipologia di approccio didattico, l'istruzionismo.

La **filosofia istruzionista** si basa sulla convinzione che la conoscenza e le competenze debbano essere trasmesse dall'insegnante allo studente, il quale assume una posizione piuttosto passiva in un contesto dove altri strumenti – ad esempio il computer – non sono altro che un semplice supporto per l'apprendimento.

La **teoria costruttivista** di Piaget, al contrario, si incentra soprattutto su una visione di scuola dove i discenti sono attori attivi e partecipativi nel loro processo di apprendimento, poiché costruiscono e raffinano in modo autonomo e incrementale le proprie competenze. In particolare Papert apprezzò moltissimo il lavoro del suo mentore sul collegamento intrinseco che, secondo Piaget, esiste tra l'intelligenza – o, per meglio dire, capacità cognitiva – e l'abilità dell'individuo di adattarsi efficacemente all'ambiente sociale e fisico in cui si trova. In particolar modo, Papert si dedicò a consolidare il costruttivismo di Piaget in una sua versione più specifica, il **costruzionismo** [15].

La teoria costruzionista di Papert pone l'accento non solo sul ruolo attivo dello studente, ma anche sul fatto che la "costruzione" divenga ancora più efficace in un contesto in cui l'individuo produce un risultato per lui ricco di significato e di utilità.

In questo modo, come si legge nel libro "*Mindstorms: Children, Computers, and Powerful Ideas*" [14], Seymour Papert propose un nuovo modello educativo in cui strumenti come il computer non sono più visti come "contenitori di informazioni" da cui attingere, ma come strumenti per costruire e manipolare le proprie idee e la propria fantasia.

Inoltre, in un altro suo scritto del 1993 "*Children's Machine – Rethinking school in the age of the computer* "[16], Papert citò il famoso proverbio africano "Dai un pesce ad una persona, e quella persona sarà sazia per un giorno, insegna ad una persona a pescare, e quella persona si sazierà per tutta la vita.". Il Costruzionismo si basa proprio sull'idea che il bambino impari in modo migliore "pescando" in autonomia le competenze e le informazioni di cui ha bisogno: l'istituzione scolastica ha il compito di supportarlo moralmente, psicologicamente, materialmente e intellettualmente nei suoi sforzi.

In particolare, Papert capì che, specialmente per quanto riguarda i giovani studenti, la **programmazione informatica** può essere l'attività ideale per sviluppare questo tipo di competenze. Lo studioso, infatti, era convinto che gli obiettivi di questo tipo di attività dovessero essere il ragionamento e la costruzione di un algoritmo che risolva un problema, insieme alla capacità di riconoscere e correggere gli errori. In generale, ritenne poi necessario compiere

delle operazioni di semplificazione di alcune istruzioni e concetti informatici, i quali potevano risultare troppo complessi ma anche non necessari allo scopo.

Secondo tali premesse, negli anni sessanta del secolo scorso Papert sviluppò un proprio linguaggio, **LOGO** - il cui nome deriva dal greco antico λὸγος ("parola" oppure "pensiero"). LOGO è un ambiente di programmazione visuale che permette di controllare una tartaruga, scrivendo codice e imparando a padroneggiare i concetti di base dell'informatica in modo semplice e divertente. L'ambiente di questo linguaggio, fortemente grafico ma allo stesso tempo semplice, è adatto a bambini di tutte le età, permettendo loro non solo di imparare a programmare, ma anche di sfruttare al massimo la loro **creatività e fantasia**. LOGO, inoltre, fu progettato in modo tale da permettere di costruire quei procedimenti cognitivi che permettono di individuare errori e inconvenienti e di sviluppare un ragionamento critico che aiuti a far fronte a questo tipo di situazioni, accettando tutte le soluzioni come possibili e non etichettandone alcune come sbagliate a priori. Tra le caratteristiche più importanti di LOGO, inoltre, vi è la possibilità di utilizzarlo non soltanto come supporto alle materie matematiche o scientifiche, ma anche come **strumento interdisciplinare** per la creazione di attività di carattere artistico e umanistico. Il computer diventa in questo modo strumento di espressione e di costruzione di conoscenza, e non solo un mezzo per l'apprendimento di nozioni preconfezionate. Inoltre, grazie all'elaboratore, lo studente può vedere il risultato concreto delle proprie conoscenze tradotte in algoritmi e programmi. Come anticipato, LOGO permette di controllare una tartaruga. La tartaruga in questione non è altro che un cursore grafico che, quando si muove, lascia il segno del proprio percorso dietro di sé grazie ad una "penna": in questo modo, muovendosi, la tartaruga può disegnare figure geometriche di ogni tipo. La "geometria della tartaruga" è un'area didattica creata da Seymour Papert e i suoi colleghi proprio per rendere più semplice e divertente l'apprendimento, in questo caso, della matematica e della geometria. I comandi base della geometria della tartaruga sono molto semplici: a partire da questi si possono ideare problemi e sfide di ogni tipo e di ogni grado di difficoltà, mettendo in campo conoscenze pregresse sulla materia e acquisendone di nuove. In ogni caso, LOGO, come tanti altri strumenti di programmazione visuale e non solo, costituisce una preziosa risorsa sia per gli studenti che per gli insegnanti, poiché promette un **nuovo modo di imparare**, che **insegna a pensare** [13]: un metodo sicuramente più attivo e coinvolgente, ma anche più efficace e duraturo.

### 1.2.3 Jeannette Wing

Il concetto di "pensiero computazionale" è stato poi ripreso e ampliato da **Jeannette Wing**, informatica teorica e attuale vicepresidentessa di Microsoft

Research.

Nel 2006, Wing pubblicò un articolo intitolato "*Computational Thinking*"[21]. Nell'articolo in questione, Wing parla di pensiero computazionale come "un approccio universalmente applicabile e un set di *skills* che tutti noi dovremmo imparare e usare, non soltanto gli informatici". Nello stesso scritto, fu proposto proprio questo concetto come la quarta abilità fondamentale da insegnare nelle scuole, insieme alla lettura, alla scrittura e all'aritmetica.

Infatti, secondo Wing, l'informatica non offre soltanto tecnologie straordinarie, ma anche un ***framework* intellettuale**[22] per un nuovo modo di pensare, utile a tutti. Wing scrisse: "Il pensiero computazionale coinvolge la risoluzione di problemi, la progettazione di sistemi e la comprensione del comportamento umano, attingendo dai concetti fondamentali del mondo dell'informatica.[...] Il pensiero computazionale è riformulare un problema che sembra difficile in uno che sappiamo già come risolvere, ad esempio attraverso processi di riduzione, incorporamento, trasformazione o simulazione".

*Divide et impera*: ecco la prima competenza fondamentale che il pensare in modo computazionale insegna. La **scomposizione di un problema in sotto-problemi più piccoli**, più facili da risolvere, aiuta ad esercitare un certo grado di controllo e ordine sul problema originale, e contemporaneamente, lo rende più gestibile e più facilmente comprensibile. Sulla stessa linea, nasce anche la **capacità di riconoscere e utilizzare *pattern***, ovvero soluzioni a problemi già risolti in passato che possono essere riutilizzate in nuovi contesti. Ma per poter risolvere un problema in modo efficiente, è anche necessario **saper astrarre e generalizzare**, analizzando in modo critico i dati e le informazioni a disposizione, eliminando quelle superflue e concentrandosi su quelle essenziali. Infine, Wing invita a **pensare in modo algoritmico**, ovvero a sviluppare una sequenza di passi logici e ben definiti che, se seguiti correttamente, portano alla soluzione del problema. Si alternano, dunque, momenti in cui si riflette sul problema, si analizzano i dati e si progettano soluzioni, applicando conoscenze pregresse e nuove, a momenti in cui si mette in pratica il frutto di questo lavoro mentale, eventualmente scrivendo codice e testando la soluzione proposta, correggendo eventuali errori. Pensare in modo computazionale, dunque, non ha solo a che vedere con l'essere in grado di programmare un computer: significa anche essere in grado di **pensare a più livelli di astrazione**. Jeannette Wing è convinta, come molti esperti del settore, che il pensiero computazionale non sia una competenza riservata soltanto agli informatici e alle persone di scienza. Infatti, insegnarlo a scuola non vuol dire necessariamente formare futuri programmatori o ingegneri, ma cittadini più consapevoli e competenti in un mondo sempre più tecnologico e complesso. In questo mondo, è la macchina a dover pensare come l'uomo, e non il contrario: l'essere umano è creativo e intelligente, la macchina è passi-

va; è l'utente a dover rendere la macchina interessante per poterla utilizzare al massimo delle sue potenzialità.

## 1.3 Il concetto di "Micromondo"

Fu Seymour Papert 1.2.2 per primo ad introdurre il concetto di "micromondo" nel contesto informatico ed educativo. Il micromondo di Papert nacque contestualmente all'introduzione del suo linguaggio di programmazione, LOGO, e della relativa protagonista, la tartaruga. L'ambiente di apprendimento della tartaruga fu pensato e realizzato per permettere ai bambini di ragionare su concetti e sfide di carattere matematico e geometrico: non a caso, Papert lo aveva soprannominato "provincia di Mathland", come ad indicare un piccolo mondo, confinato e protetto, costruito ad hoc con un obiettivo ben preciso, quello di fare scuola in modo assolutamente innovativo.

Secondo Papert, infatti, il micromondo nasce come un **ambiente circoscritto**, in cui è possibile **sperimentare e coltivare le proprie idee con un alto grado di libertà e autonomia**, acquisendo conoscenze in modo naturale e graduale, proprio come avviene nei primi anni di vita dell'essere umano. Generalmente, la costruzione di micromondi avviene attraverso una serie di fasi e principi ben definiti, che contribuiscono a rendere l'ambiente il più efficace e stimolante possibile.

In primo luogo, come già accennato, i micromondi si basano sempre su un tema ben preciso: il dominio in cui si svolge l'attività deve fare in modo di spostare l'attenzione degli studenti sul mettere in pratica le conoscenze acquisite e i loro personali modi di pensare. Naturalmente, il tema scelto deve essere non solo facilmente comprensibile da chi lo programma, ma deve lasciare spazio alla creatività e alla fantasia. Papert stesso scrisse: ***"Low floors, high ceilings"***. *Low floors*, traducibile come "pavimenti bassi", richiama la facilità di accesso e di comprensione che ogni micromondo deve garantire: l'ambiente deve essere alla portata di tutti, senza discriminazioni di alcun tipo. *High ceilings*, ovvero "alti soffitti", invece, si riferisce alla possibilità di oltrepassare i limiti imposti dall'ambiente stesso e di dare spazio alla componente fantasiosa e creativa di ogni studente; in tal modo le possibilità di realizzazione del micromondo sono pressoché infinite. Successivamente l'espressione venne ampliata da Mitchel Resnick, uno degli allievi di Papert e attualmente professore presso il MIT di Boston, con l'aggiunta di "*wide walls*", ovvero "mura larghe". Resnick si riferiva alla possibilità di estendere il micromondo includendo concetti e sfide sempre nuove e attuali, facendo in modo di poter continuare a raccogliere contributi e idee da parte di tutti coloro che hanno accesso ad esso.

In questo modo, si rende possibile la creazione di contesti didattici stimolanti e immersivi, dove **l'aspetto ludico e quello educativo si fondono in modo armonioso e naturale**, permettendo agli studenti di sentirsi personalmente coinvolti e motivati.

Tutto ciò, nondimeno, non sarebbe possibile senza il coinvolgimento attivo degli insegnanti, che devono essere in grado sia di gettare le basi per la costruzione del micromondo, sia di supportare e guidare l'alunno nel suo percorso di apprendimento, che risulta significativamente più autonomo e personale rispetto a quello tradizionale. Il docente, in questo nuovo modo di fare scuola, assume il ruolo di guida e di facilitatore, piuttosto che di dispensatore di nozioni e conoscenze: deve necessariamente lasciare spazio a tutte le idee degli studenti, accettando e valorizzando anche il ruolo degli "**errori**" come parte integrante del processo di apprendimento.

Non solo: gli studenti possono sperimentare un approccio più attivo e partecipativo rispetto alla didattica frontale. incrementale, le loro idee e i loro ragionamenti, producendo un risultato significativo e soprattutto tangibile. In particolare, i risultati più importanti di questo metodo educativo si ritrovano nello sviluppo di una lunga serie di **competenze trasversali**, o *soft skills*. Da una maggiore capacità di *problem solving* e pensiero critico, a una più sviluppata autonomia e responsabilità, non dimenticando un importante sviluppo nella capacità di collaborazione e di comunicazione se si lavora in gruppo, il micromondo si rivela un'ottima occasione di crescita e di apprendimento per gli studenti di ogni età.

## 1.4 Il *Coding*

I concetti emersi nei paragrafi precedenti aprono le porte a un'ulteriore riflessione, quella riguardante gli strumenti e le metodologie corrette per trasmettere la *forma mentis* del pensiero computazionale.

L'informatica possiede una dignità intrinseca quale disciplina scientifica fondamentale, caratterizzata da un corpus consolidato di concetti e metodologie indipendenti. Inoltre, riveste un valore significativo anche come **disciplina trasversale**, offrendo un approccio che contribuisce a una più profonda comprensione delle altre discipline. Le "Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione[2]", pubblicate dal MIUR nel 2012, sottolineano come la scuola italiana debba evolversi, permettendo alle nuove generazioni di sfruttare le competenze base – che, in passato, erano l'obiettivo primo del processo di alfabetizzazione – come strumenti per una miglior comprensione del contesto sociale in cui è inserito lo studente.

[2] `https://www.miur.gov.it/documents/20182/51310/DM+254_2012.pdf`

Ken Robinson, nella prefazione allo scritto "*Lifelong Kindergarten - Cultivating Creativity through Projects, Passion, Peers, and Play*"[17] di Mitchel Resnick, riflette sul fatto che la tecnologia ha sempre assistito l'uomo nell'espansione del proprio corpo e della propria mente. La capacità umana di **astrazione e immaginazione** è distintiva e fondamentale per l'evoluzione della società e della cultura. Tuttavia, per poter alimentare queste qualità, è necessario disporre di strumenti e materiali adatti. Poiché l'informatica consente di costruire rappresentazioni e soluzioni per situazioni anche fisicamente irrealizzabili, limitate solo dalla nostra immaginazione, essa si presenta come un ambiente potente, utile per esercitare la creatività, assumendo anche un elevato valore educativo. In un contesto in cui le informazioni sono facilmente accessibili e i computer sono sempre più abili nel risolvere compiti noiosi e ripetitivi, diventa ancor più cruciale **educare gli studenti a trovare soluzioni innovative**, anziché concentrarsi sull'applicazione meccanica di procedure mnemoniche o standardizzate.

Resnick, a tal proposito, sottolinea che il concetto di creatività viene associato principalmente all'espressione artistica o alla "Creatività con la C maiuscola" di grandi inventori e artisti le cui opere decorano i musei. La creatività, però, non appartiene solo agli artisti, ma anche agli scienziati che fanno scoperte, ai medici che diagnosticano malattie e trovano nuove cure, o agli imprenditori che sviluppano nuovi prodotti o strategie di mercato. In modo più generale, la creatività entra in gioco quando si generano idee utili per sé stessi o per gli altri nella vita quotidiana, non necessariamente "rivoluzionarie" a livello globale, ma sicuramente significative a livello personale. In altre parole, si tratta di una qualità alla portata di tutti, purché sia coltivata fin dalla giovane età, ancora meglio se nei contesti educativi e scolastici.

Se si pensa al mondo dell'informatica, dunque, il *coding*, ovvero la scrittura di codice, si configura come uno strumento di straordinaria potenza nell'ambito dello sviluppo del pensiero computazionale: si tratta di un profondo tessuto di competenze che vanno ben oltre la mera scrittura di algoritmi e istruzioni.

In primo luogo, la programmazione agisce come **catalizzatore del pensiero critico**, spingendo gli studenti a frazionare intricati problemi in componenti più gestibili, fornendo loro un approccio strutturato nella risoluzione delle sfide. Quando inserito in contesti di piccoli gruppi di lavoro, il *coding* favorisce **la collaborazione e il sinergico lavoro di squadra**, sottolineando l'importanza della condivisione di idee e della cooperazione per raggiungere obiettivi comuni. La **componente creativa** della programmazione emerge in modo preminente: risolvere un problema non richiede solo la padronanza di conoscenze pregresse, ma anche la capacità di **immaginare e progettare soluzioni**, valutandone attivamente i pro e i contro e selezionando la strategia più idonea. In questo contesto, il *coding* diventa un veicolo per l'espressione

personale e il confronto con diverse prospettive, aprendo a un ventaglio infinito di possibilità creative. Infine, al di là delle competenze tecniche specifiche, il *coding* prepara gli studenti per un futuro digitale in continua evoluzione, potenziando la loro **alfabetizzazione tecnologica** e aprendo porte a innumerevoli opportunità nel vasto campo della tecnologia. In sintesi, il *coding* non solo istruisce sul linguaggio della programmazione, ma plasma **menti agili, creative e collaborative**, pronte a navigare le sfide della società digitale moderna.

Tuttavia, l'introduzione delle discipline informatiche nel contesto educativo non è un'operazione immediata né tantomeno semplice. Al giorno d'oggi vi sono ancora pregiudizi e critiche che circondano l'utilizzo - spesso eccessivo e improprio - della tecnologia da parte dei giovani.

Una possibile sfida, ad esempio, riguarda l'associazione concettuale che vede l'informatica come un'attività principalmente giocosa e ludica: se da un lato è vero che il *coding* può essere un'attività divertente e stimolante, dall'altro è importante sottolineare che la programmazione è una disciplina seria e complessa, che richiede molto impegno e dedizione.

Un altro possibile ostacolo, diametralmente opposto per quanto riguarda ciò di cui si è parlato poc'anzi, risiede nei luoghi comuni e nei pregiudizi che ancora oggi circondano la disciplina. Idee sbagliate e stereotipate vedono attività quali programmare e ragionare da informatici come appannaggio di poche persone che hanno quello che gli americani chiamano il "gene del *nerd*" (*Geek gene*) [1] spesso associato a personalità asociali e competitive e, più in generale, ad un target prevalentemente maschile.

In ultimo, vi sono anche delle difficoltà di natura pratica, legate alla formazione degli insegnanti e alla disponibilità di strumenti e risorse adeguate.

La formazione degli insegnanti, infatti, è un aspetto cruciale per garantire il successo di un'iniziativa di questo tipo: non solo è necessario che gli insegnanti siano in grado di padroneggiare le competenze tecniche necessarie per insegnare il *coding*[8], ma è anche importante che siano in grado di integrare queste competenze in un contesto educativo più ampio, in modo da garantire che il *coding* non sia un'attività isolata, ma che si integri in modo armonioso con le altre materie e discipline.

Per quanto riguarda, invece, la possibilità di avere accesso a tecnologie e strumenti utili per questa tipologia di insegnamento, bisogna considerare che non tutte le scuole del mondo sono in grado di permettersi di acquistare i materiali necessari. In aree più svantaggiate, ad esempio, dove è già difficile garantire l'accesso a materiali didattici di base, l'idea di introdurre il *coding* come disciplina può sembrare un lusso eccessivo. Tuttavia, moltissimi studi hanno dimostrato che l'uso di strumenti tecnologici non è una *conditio sine qua non* per insegnare la programmazione informatica. Esistono infatti meto-

dologie molto efficaci e, al contempo assolutamente economiche, che riescono a trasmettere il pensiero computazionale anche senza l'uso del computer. Il *coding* può essere insegnato anche attraverso attività di *role playing*, di *problem solving* e di *game sbased learning* dette *unplugged*. Un esempio di questi studi è stato compiuto da Arinchaya Threekunprapa e Pratchayapong Yasri [20], che hanno dimostrato come l'uso di *unplugged coding* usando dei *flowblocks* abbia portato a risultati ottimi per quanto riguarda la promozione del pensiero computazionale tra studenti della scuola secondaria. Il *paper*, pubblicato nel 2020, mette in luce come, anche utilizzando delle semplici forme di carta - rappresentanti i diversi blocchi di codice e i relativi significati algoritmici - si possa ottenere un apprendimento significativo e duraturo. L'attività proposta era ***game based***, ovvero posta in un contesto ludico e divertente, e ha portato a risultati molto positivi, sia in termini di apprendimento che di coinvolgimento degli studenti. I partecipanti, lavorando in coppia, non solo hanno dovuto collaborare e comunicare tra loro per risolvere le diverse sfide, ma hanno avuto anche un supporto morale e psicologico nel momento in cui si sono trovati ad affrontare sfide sempre più difficili, incontrando anche errori e fallimenti. Il confronto, però, ha reso la fase del *debugging* molto più semplice e veloce, e, insieme anche ad una buona dose di competitività tra i vari gruppi, tutti i partecipanti sono riusciti a portare a termine con successo le sfide proposte. La fase finale della sperimentazione di Threekunprapa e Yasri ha visto i partecipanti - che non avevano alcuna esperienza pregressa di programmazione - migliorare significativamente rispetto al test iniziale.

# Capitolo 2

# Strumenti per la creazione di micromondi digitali: Snap! e CoSpaces Edu

Nel corso della fase sperimentale (descritta dettagliatamente nel Capitolo 3), diversi strumenti sono stati impiegati per sviluppare applicazioni e attività di programmazione visuale. **Snap!**, in particolare, si è dimostrato particolarmente efficace nell'introduzione dei concetti fondamentali della programmazione e nella costruzione della comprensione della **logica a blocchi**.

Tuttavia, quando l'obiettivo era la creazione di **applicazioni più coinvolgenti e immersive**, i docenti coinvolti hanno preferito adottare **CoSpaces Edu**. Questo strumento offre un ambiente di programmazione visuale che consente la creazione di progetti tridimensionali fruibili anche attraverso l'esperienza della realtà virtuale.

In questo capitolo si vogliono presentare gli strumenti citati per darne una panoramica generale e per mostrare come essi siano stati utilizzati per la realizzazione delle attività didattiche.

## 2.1 Snap!

Snap! - formalmente *Build Your Own Blocks* (BYOB) - è un ambiente di programmazione visuale sviluppato dal MIT Media Lab. Nato sulla falsariga di Scratch, come quest'ultimo anche Snap! è basato principalmente sulla sintassi dei **blocchi** e si propone di rendere l'apprendimento della programmazione accessibile e divertente.

Programmare "a blocchi" si riferisce a un approccio di programmazione visuale in cui il codice del programma è rappresentato graficamente sotto forma di blocchi o "mattoncini" (ciascuno dei quali rappresenta una specifica

operazione, funzione o comando), che vengono assemblati tra loro per creare sequenze di istruzioni, cicli, condizioni e così via.

Il programma è ***open source*** e può essere utilizzato gratuitamente da qualunque browser. Si ritrova anche una vastissima comunità online, che pubblica e condivide risorse utili e interessanti come progetti, tutorial, esempi. Dunque, Snap! si conferma come uno strumento molto **colorato, accattivante, nonché visivamente intuitivo**.

Snap! è stato sviluppato per essere utilizzato da un pubblico giovane e da principianti. Tuttavia, offre anche funzionalità avanzate che lo rendono adatto a progetti più complessi e all'insegnamento di concetti informatici avanzati come, ad esempio, la programmazione ad oggetti. Inoltre, Snap! è stato progettato per essere un ambiente di programmazione flessibile e modulabile, consentendo agli utenti di creare e personalizzare i propri blocchi e di utilizzare un vasto numero di librerie aggiuntive (tra le quali anche MQTT) e risorse esterne disponibili online.

### 2.1.1 Interfaccia utente

L'interfaccia utente di Snap! è composta da un **canvas** principale su cui gli utenti posizionano e collegano i blocchi (secondo la modalità *drag-and-drop*), e da una serie di pannelli laterali che forniscono accesso a vari strumenti e risorse.

Sulla parte alta dello schermo è presente una barra dei menu che consente di accedere alle funzionalità di base, come la creazione di nuovi progetti e l'apertura di lavori già esistenti, la condivisione di programmi online e così via.

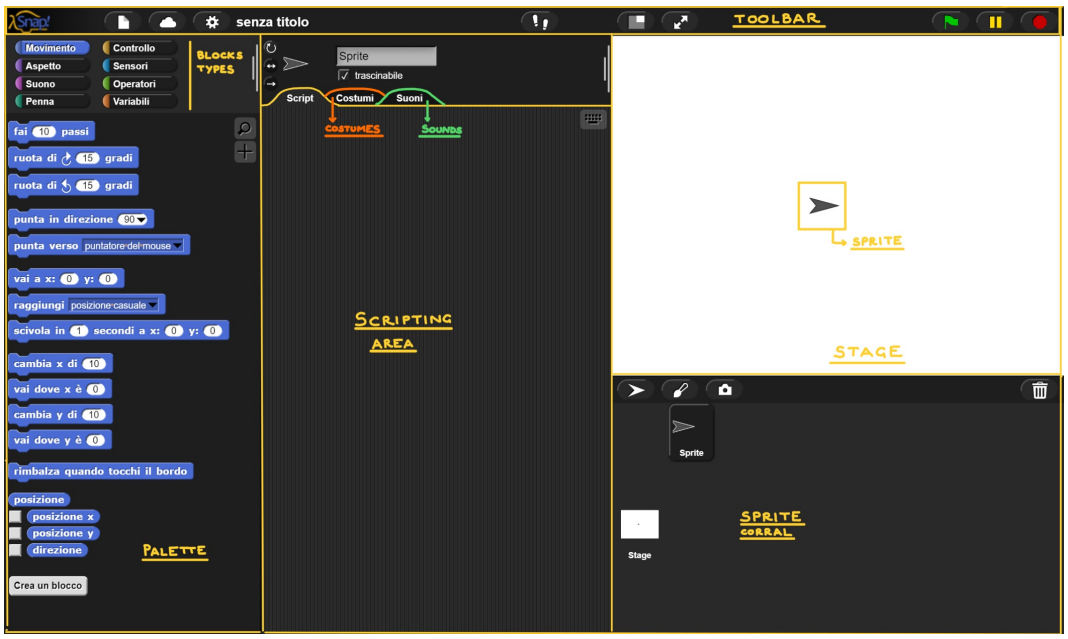



Figura 2.1: Interfaccia utente di Snap!.

Come si può vedere dalla figura 2.1, l'interfaccia utente di Snap! è divisa in tre macro-aree principali che dividono la schermata del browser.

Partendo da sinistra, si trova la **palette dei blocchi**, che contiene tutti i blocchi disponibili per la programmazione. Ve ne sono di diversi tipi: ogni categoria di blocco è contrassegnata da una certa forma e da un colore specifico, al fine di rendere più facile la ricerca e l'identificazione dei blocchi desiderati. Le principali categorie di blocchi includono:

- **Movimento**: per controllare il movimento e la posizione degli *sprite*;
- **Aspetto**: per modificare l'aspetto degli *sprite*, come la loro dimensione, i costumi, aggiungere vignette per i dialoghi e i pensieri e così via;

- **Suono**: per controllare la riproduzione dei suoni;
- **Penna**: che permettono di "disegnare" sullo stage - lo *sprite* funge da cursore grafico e, ogni volta che si muove, lascia una traccia dietro di sé;
- **Controllo**: per controllare il flusso del programma, come i cicli e le condizioni;
- **Sensori**: per interagire con l'ambiente esterno, per eseguire operazioni come il rilevamento del movimento, della posizione del mouse e così via;
- **Operatori**: per eseguire operazioni matematiche, logiche e di confronto;
- **Variabili**: per creare e utilizzare variabili;

Inoltre, come già anticipato, per ogni categoria di blocchi è possibile crearne di nuovi e personalizzarli a proprio piacimento.

La sezione centrale, detta anche **scripting area**, è il luogo in cui gli utenti posizionano e collegano i blocchi per creare script e programmi. Gli utenti possono trascinare i blocchi dalla palette e collegarli insieme per creare sequenze di istruzioni, cicli, condizioni e così via. I blocchi, grazie alla loro particolare forma, seguono regole di incastro ben precise, che ne permettono il corretto collegamento e l'identificazione del loro scopo: da qui il nome, "*snap*" che richiama lo "schiocco" che si sente quando due pezzi di costruzioni (o puzzle) si incastrano correttamente.

Sempre nella sezione centrale, è possibile modificare l'aspetto degli *sprite* con dei costumi - immagini che rappresentano lo *sprite* in un certo stato - e aggiungere suoni per rendere l'esperienza più coinvolgente.

Infine, nella parte più a destra dell'interfaccia grafica si trova lo **stage** vero e proprio, ovvero l'area in cui gli *sprite* si muovono e interagiscono tra loro. Gli *sprite* possono essere spostati e ridimensionati direttamente sullo stage, e gli utenti possono interagire con essi tramite il mouse e la tastiera. In basso, inoltre, è presente una barra di navigazione che consente di passare da uno script (inteso come "scena") all'altro e la lista completa degli *sprite* presenti nel progetto.

### 2.1.2 Programmare con Snap!

Snap! offre un'ampia gamma di funzionalità di programmazione, che vanno dalla gestione degli eventi e del flusso del programma, alla manipolazione dei dati e alla creazione di procedure personalizzate. Gli utenti possono creare infinite combinazioni di blocchi per creare script complessi e interattivi, e, al contempo, imparare i concetti fondamentali della programmazione, come le

variabili, i cicli, le condizioni, le funzioni e così via. Snap! integra anche liste e *array* per gestire i dati in modo strutturato.

Un primo esempio di script molto semplice è mostrato in figura 2.2.

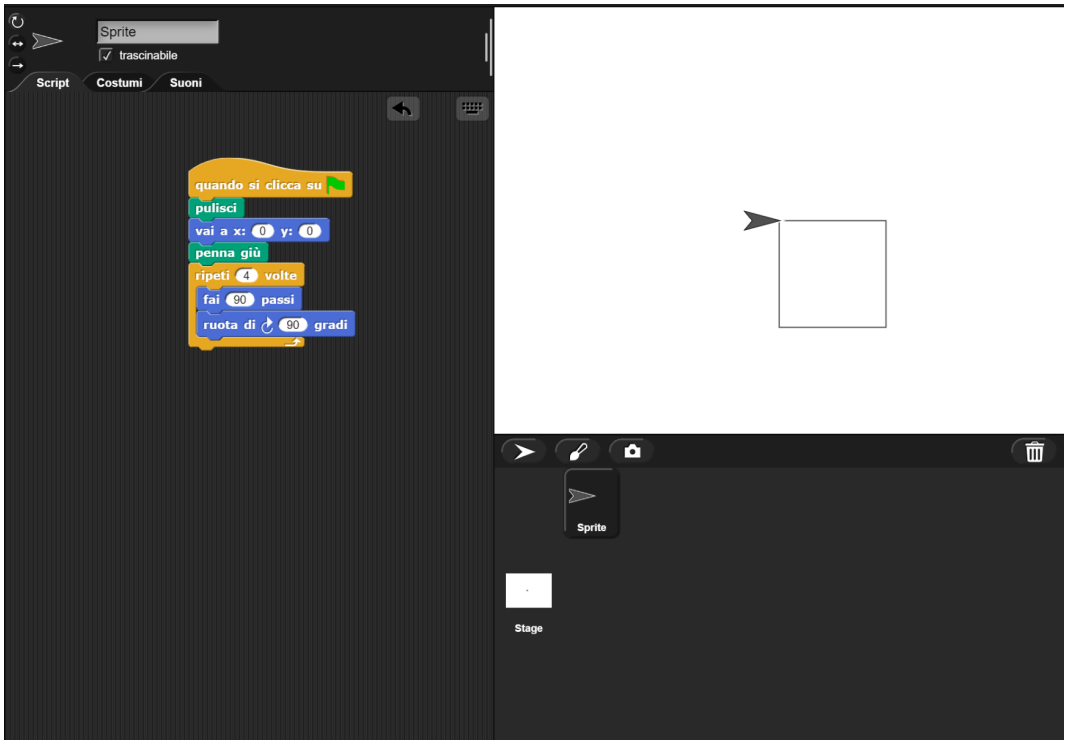



Figura 2.2: Esempio di script in Snap!.

Come si può notare, lo script è introdotto da un blocco 

(un blocco **hat** di Controllo), che ne definisce l'evento di attivazione: ve ne sono di diversi tipi, proprio per rispondere a diversi eventi (e.g. la pressione di un determinato tasto o la ricezione di un messaggio da parte di un altro script).

Lo script in questione è molto semplice: dopo aver appoggiato la penna (sullo stage) sposta lo *sprite* in avanti di 90 passi, lo ruota di 90 gradi e ripete quest'operazione 4 volte. Il risultato, come si vede a destra nella figura 2.2, è un quadrato disegnato sullo stage. Questo primo esempio, molto semplice, coordina sia la conoscenza della figura geometrica del quadrato, sia la programmazione di base. In particolare, per scrivere questo programma, l'utente deve conoscere le caratteristiche della figura che sta disegnando: in questo caso, il quadrato ha 4 lati uguali e 4 angoli retti. Lo step successivo è quello di comprendere che l'operazione "muovi avanti" e "ruota" devono essere ripetute

4 volte per ottenere la figura desiderata, senza dover ripetere più volte le stesse istruzioni: questo è un esempio di ciclo, un concetto fondamentale della programmazione e facilmente riproducibile in Snap! con il blocco 

. Per di più, con l'utilizzo dei cicli, gli alunni possono sperimentare e vedere con i loro occhi la semplificazione del codice, che diventa più leggibile e più facile da modificare in futuro. Chiaramente, l'obiettivo finale è quello di ricercare soluzioni sempre più efficienti e semplici da comprendere.

Snap!, dunque, si configura come un ottimo strumento sia per l'introduzione alla programmazione, sia per l'apprendimento di concetti più avanzati. Inoltre, Snap! permette una ottima organizzazione del codice, grazie alla possibilità di creare e utilizzare variabili e funzioni personalizzate, e di organizzare il programma in script separati: il processo di debugging e di manutenzione del codice risulta quindi molto più agevole e intuitivo. Indubbiamente la sua interfaccia grafica e la sua facilità d'uso lo rendono coinvolgente e motivante per gli studenti, che possono vedere immediatamente i risultati dei loro programmi e che riescono a creare micromondi straordinari utilizzando sfondi e personaggi da loro disegnati. Le possibilità sono infinite: si possono creare giochi, storie interattive, animazioni, simulazioni e molto altro ancora. Sicuramente questo software pone le basi per un apprendimento costruzionista - proprio come sostiene la teoria di Papert, per l'acquisizione del pensiero computazionale e di tutte le meta-competenze ad esso correlate.

## 2.2 CoSpaces Edu

CoSpaces Edu è un'applicazione che consente di creare micromondi interattivi e di programmarli con un linguaggio di programmazione visuale. È disponibile sia come applicazione web che come applicazione mobile, in modo tale da poter essere utilizzata su diversi dispositivi e da diversi utenti.

La fruizione di CoSpaces Edu è gratuita, anche se la versione base è alquanto limitata e non consente di accedere a tutti i blocchi utilizzabili nella scrittura degli script. È possibile sottoscrivere un abbonamento annuale per accedere a funzionalità aggiuntive e a contenuti premium, ma ad un costo non indifferente.

CoSpaces Edu mette insieme diverse aree della tecnologia e dell'informatica estremamente all'avanguardia: infatti, è possibile introdurre bambini e ragazzi al mondo della **programmazione**, della **realtà virtuale** e della **realtà aumentata**, il tutto in modo semplice e avvincente.

In particolare, grazie ad una serie di strumenti di modellazione 3D, è possibile creare ambienti virtuali popolati da oggetti tridimensionali, e programmarli con un linguaggio di programmazione visuale a blocchi molto simile a quello visto per quanto riguarda Snap!.

Inoltre, CoSpaces Edu offre la possibilità di creare applicazioni di *extended reality* e realtà virtuale - esplorabili tramite visori VR - e di realtà aumentata, visualizzabili tramite dispositivi mobili come tablet e smartphone oppure con l'ausilio di opzioni *add-on* come *Merge Cube*[3].

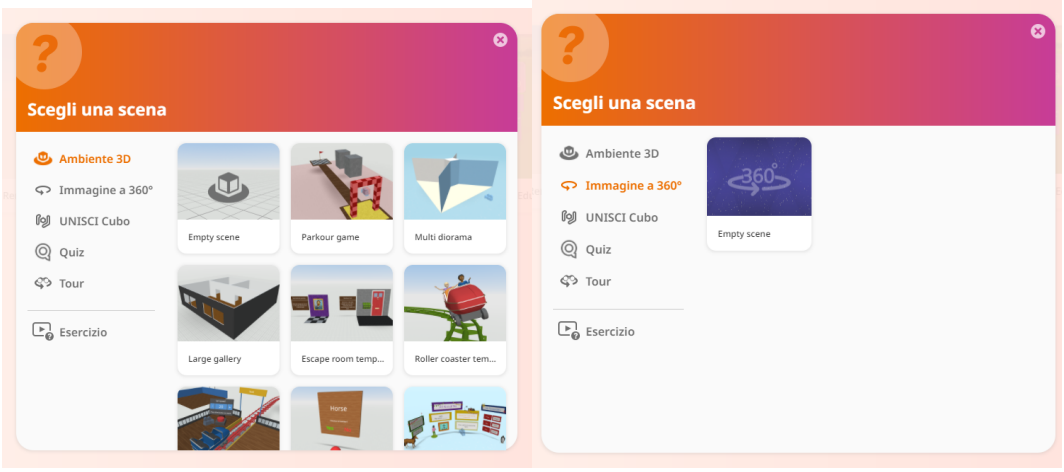
Il micromondo, dunque, diviene un'esperienza molto più coinvolgente e immersiva per gli studenti, che possono esplorarlo e interagire in prima persona con gli oggetti e gli ambienti che essi stessi hanno creato. Anche in questo caso vi sono infinite possibilità di utilizzo: si possono creare storie interattive, giochi, simulazioni, tour (a 360°) e molto altro ancora. Inoltre, CoSpaces si presta benissimo anche per la creazione di progetti interdisciplinari, specialmente in ambito STEM, ma anche per la creazione di progetti artistici, letterari e di carattere sociale.

Un'altra caratteristica che distingue CoSpaces Edu è il fatto che, nonostante sia uno strumento pensato per gli studenti, offre anche funzionalità avanzate per gli insegnanti, che possono creare e gestire classi virtuali, assegnare compiti e monitorare i progressi degli studenti. Inoltre, come Snap!, CoSpaces Edu conta una grandissima comunità online, che offre una vasta libreria di risorse didattiche, come tutorial, esempi, progetti e lezioni.

[3] `https://www.cospaces.io/merge-cube`

### 2.2.1 Interfaccia utente

Al momento della creazione di un nuovo progetto, l'utente, prima di tutto, deve scegliere il tipo di scena che vuole creare. Vi sono diverse opzioni tra cui scegliere: nel contesto di questa tesi, si sono utilizzati principalmente questi due tipi di scene:




(a) Ambiente 3D (b) Tour 360°

Figura 2.3: Opzioni utilizzate per la creazione dei progetti CoSpaces.

Per quanto riguarda queste due scene in particolare, la prima consente di creare un ambiente 3D in cui è possibile spostarsi nello spazio all'interno del micromondo (specialmente in realtà virtuale); nel secondo caso, invece, si tratta di un tour a 360°, ovvero un'immagine panoramica che permette la visualizzazione dell'ambiente in VR soltanto ruotando su se stessi, senza la possibilità di muoversi. Per entrambe le tipologie di scena, è possibile aggiungere oggetti 3D, personaggi, suoni, luci ed effetti speciali dalla libreria offerta dal software.

In ogni caso, una volta scelta la tipologia di scena, l'utente si trova di fronte ad un'interfaccia grafica molto simile a quella di Snap!, con la differenza che, all'interno di questo programma il concetto di spazialità è incentrato sulla tridimensionalità e su una profondità di campo che non si riscontra in Snap! (che, come già detto, lavora con ambienti bidimensionali).

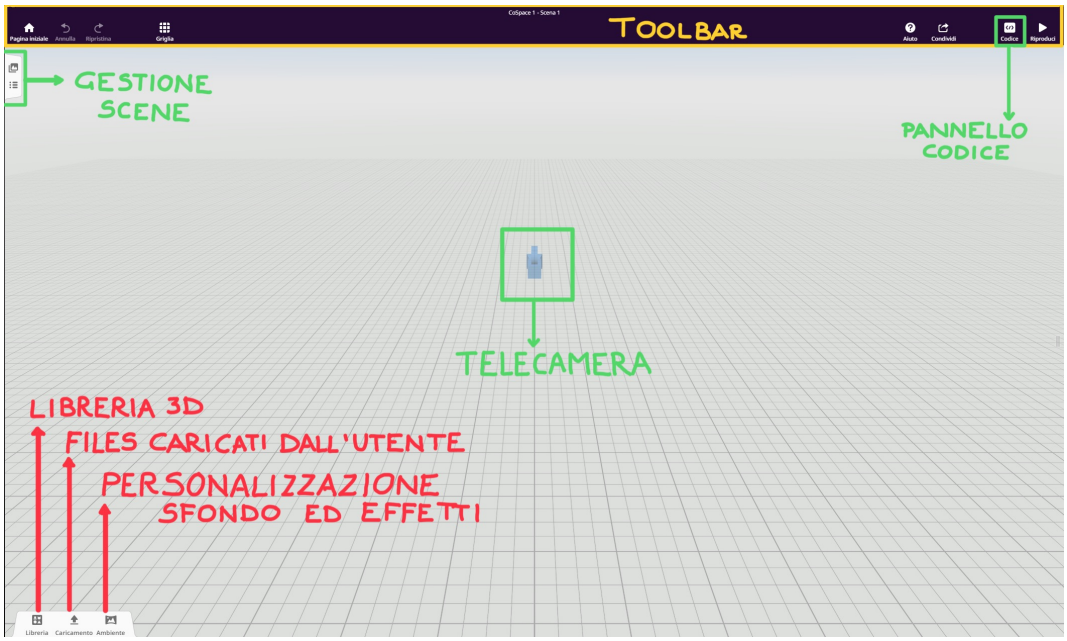
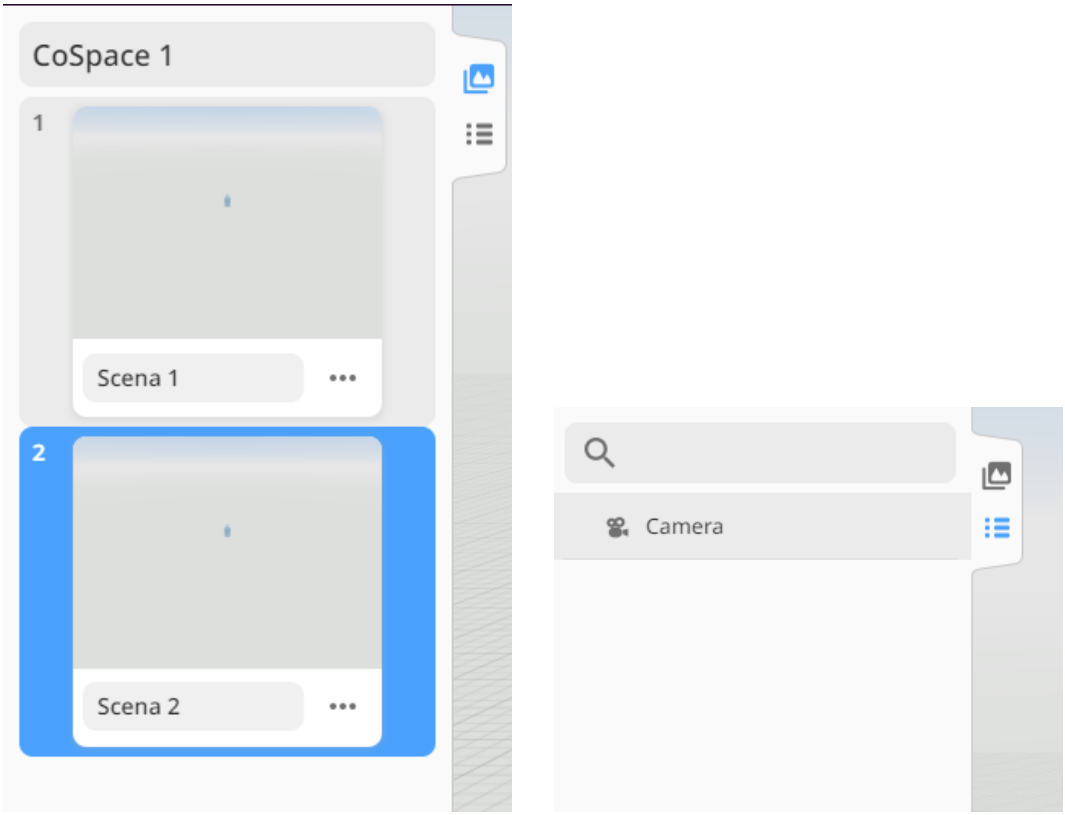



Figura 2.4: Interfaccia utente di un progetto vuoto.

In figura 2.4 è mostrata l'interfaccia utente della scena 3D di CoSpaces Edu. Il protagonista della scena è proprio lo spazio, che si configura come un'area di lavoro tridimensionale in cui gli utenti possono posizionare (anche in questo caso, in modalità *drag-and-drop*) gli oggetti del micromondo.

All'apertura della scena, l'area di lavoro è vuota, tranne che per la presenza di una telecamera posizionata al centro: questa rappresenta il punto di vista dell'utente, ovvero la sua prospettiva quando si inizierà ad esplorare il micromondo con l'utilizzo del visore oculare. In alto è presente una barra strumenti molto semplice e intuitiva.

Sulla sinistra, si trovano due icone che aprono un menù laterale a comparsa. La prima icona apre la lista delle diverse scene del micromondo, che possono essere aggiunte, modificate e cancellate a piacimento. Al momento della creazione di una nuova scena, l'utente, come già visto in precedenza nella figura 2.3, può compiere di nuovo la scelta del tipo di ambiente da creare; in questo modo, ogni micromondo può essere composto da un insieme di ambientazioni diverse. La seconda icona, invece, apre la lista degli oggetti presenti nello spazio, mostrandone la posizione corrente e permettendo di modificarne le proprietà (per esempio, nascondendoli alla vista).




(a) Scene (b) Oggetti

Figura 2.5: Menù laterale.

In basso, invece, sono presenti tre icone che aprono, anch'esse, dei menù a comparsa. La prima icona permette l'accesso alla libreria proprietaria di CoSpaces Edu, che contiene una vasta gamma di oggetti 3D, con personaggi, alloggi, animali, effetti speciali e tanto altro ancora.

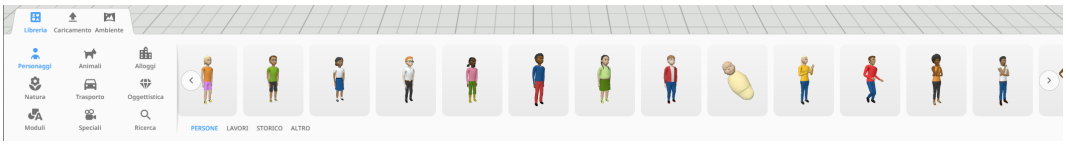


Figura 2.6: Libreria di oggetti di CoSpaces Edu.

La seconda icona, invece, mostra tutti i file caricati dall'utente: questi possono essere immagini, suoni, video e addirittura modelli 3D esterni. Questa funzionalità è fondamentale per dare carattere e personalità al micromondo, che, altrimenti, sarebbe composto solo da oggetti standard e generici.

Infine, l'ultimo pulsante del menù in basso permette la personalizzazione dell'ambiente della scena. È possibile aggiungere luci particolari, un suono di sfondo, un ambiente preimpostato (foreste, città, ambienti casalinghi, ecc.) ed effetti naturalistici (pioggia, neve, ecc.).

In ultimo, in alto a destra è presente un pulsante che apre l'area di programmazione visuale, in cui è possibile scrivere script per controllare gli oggetti della scena.

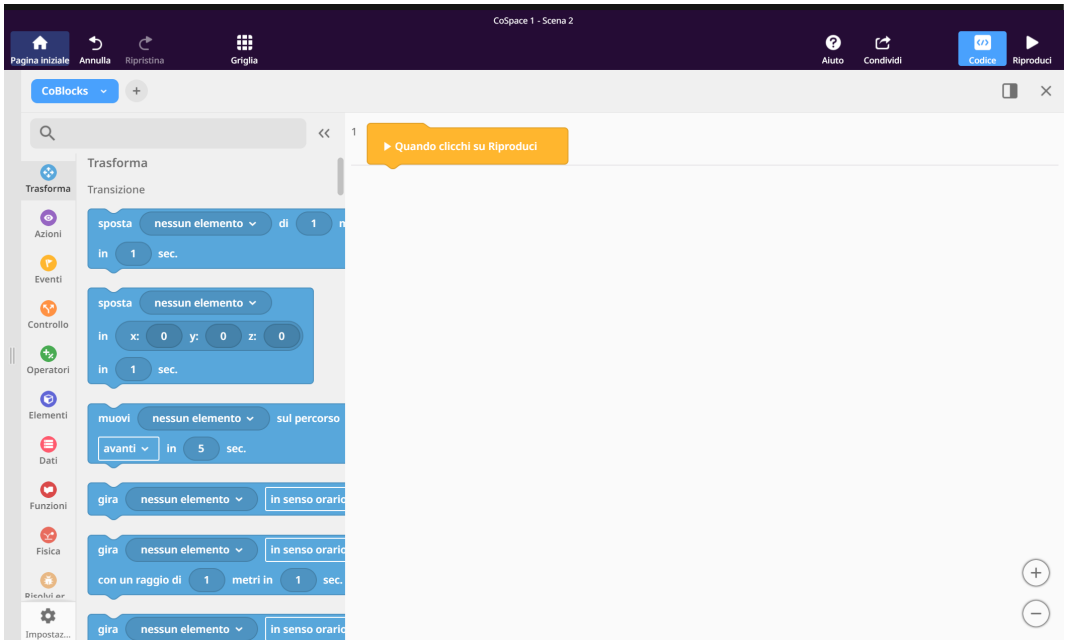


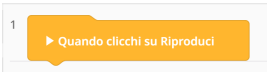

Figura 2.7: Area di programmazione visuale di CoSpaces Edu.

Come si può vedere dalla figura 2.7, l'area di programmazione è del tutto simile a quella vista in Snap! e in altri ambienti di programmazione visuale. Anche in questo caso, i blocchi sono divisi per categorie, con forme e colori precisi:

- **Trasforma**: per controllare lo spostamento degli oggetti, la loro posizione, rotazione e la loro scala;
- **Azioni**: che controllano le azioni degli oggetti, come l'attivazione di animazioni e dialoghi, la riproduzione di suoni e video e la possibilità di proporre agli utenti dei quiz interattivi;
- **Eventi**: per l'attivazione e la gestione degli eventi, come il caricamento della scena, l'interazione con gli oggetti e così via;

- **Controllo**: per controllare il flusso del programma, come i cicli e le condizioni;
- **Operatori**: per eseguire operazioni matematiche, logiche e di confronto;
- **Elementi**: per la gestione degli oggetti presenti sulla scena e delle loro proprietà;
- **Dati**: per creare e utilizzare variabili (e liste);
- **Funzioni**: per creare e utilizzare funzioni personalizzate;
- **Fisica**: per la gestione delle proprietà fisiche degli oggetti, come la gravità, la collisione e così via.
- **Risolvi errori**: per scrivere commenti e per la stampa di valori o messaggi per il debugging.

Si ritiene importante specificare che, nella versione gratuita del software, non tutti i blocchi elencati sono accessibili.

Tutti gli script presentano un blocco **hat** fisso: 

. Dunque, all'avvio del programma con il bottone in alto a destra, tutti gli script della scena vengono eseguiti. Per aggiungere altri script - che in CoSpaces vengono chiamati "CoBlocks" - alla scena, occorre soltanto cliccare sul pulsante . Grazie ai moduli CoBlocks è possibile programmare, come suggerisce il nome, a blocchi. CoSpaces, però, consente anche di scrivere codice in JavaScript (più precisamente in TypeScript) e in Python (una *feature* che, attualmente, è in fase di *beta testing*) per gli utenti più esperti e per la costruzione di progetti più complessi.

### 2.2.2 Programmare con CoSpaces Edu

In CoSpaces Edu, come in Snap!, sono disponibili moltissimi blocchi per creare script di ogni tipo e complessità. Gli utenti possono creare sequenze di istruzioni, con l'utilizzo di cicli, condizioni, variabili, funzioni e altro ancora, e possono personalizzare gli oggetti della scena in base a determinati eventi e interazioni. Anche in questo caso si vuole presentare un esempio di programma molto semplice per illustrare le potenzialità di questo strumento.

Per prima cosa, è consigliabile preparare adeguatamente la scena, scegliendo l'ambiente più adatto e posizionando gli oggetti che lo popolano. Nell'esempio in figura 2.8, si è scelto di creare un ambiente 3D un personaggio che risponde al click del mouse su di esso.

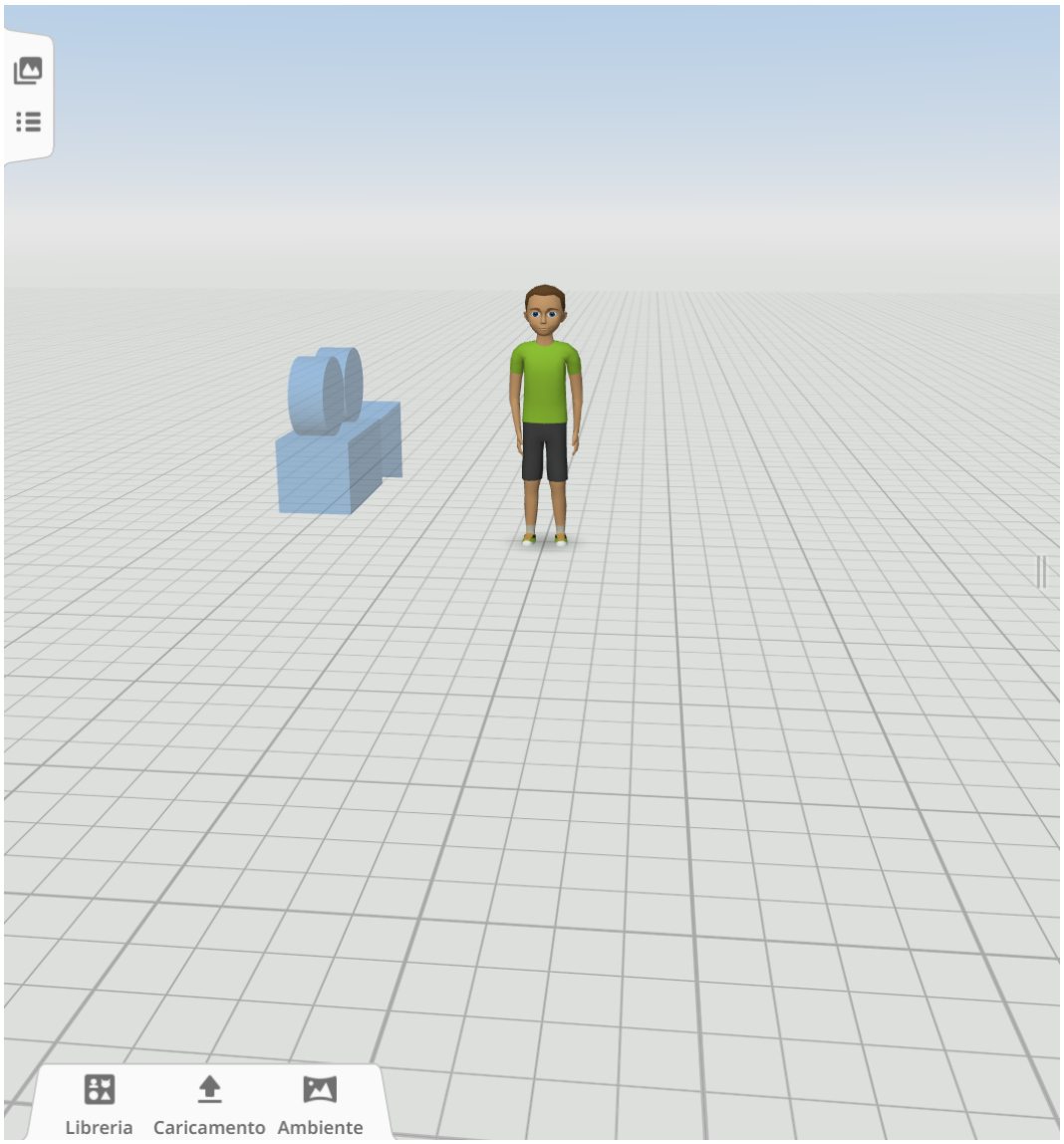



Figura 2.8: Preparazione della scena 3D

Una volta preparata la scena, si può passare alla programmazione vera e propria. Per poter utilizzare i diversi oggetti nella programmazione a blocchi, è necessario attivare l'opzione "Utilizza in CoBlocks" come mostrato in figura 2.9.

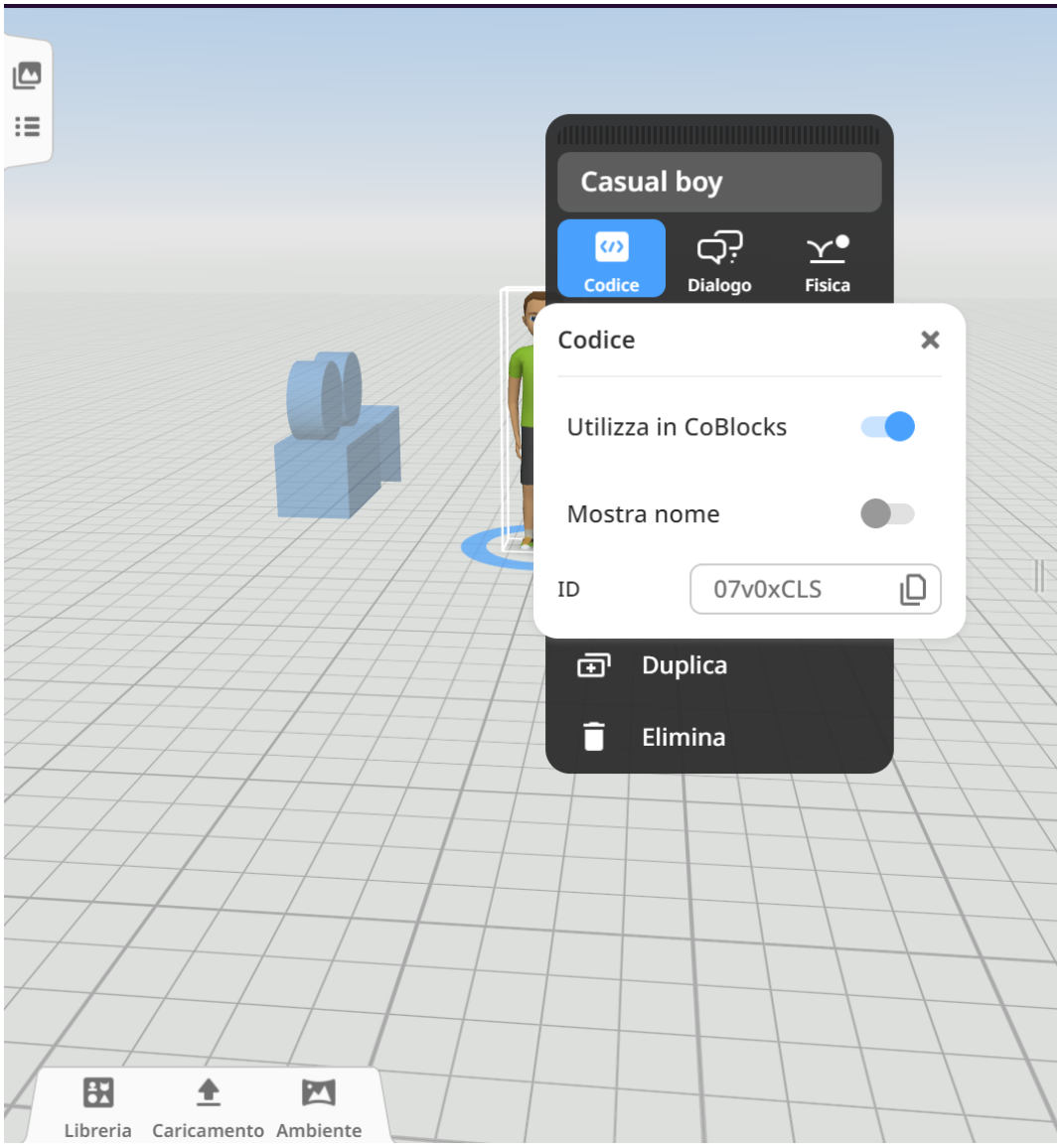



Figura 2.9: Attivazione dell'opzione "Utilizza in CoBlocks"

In questo modo, all'oggetto viene assegnato un preciso ID che lo identifica univocamente all'interno della scena e del programma. A questo punto, si può passare alla creazione dello script vero e proprio, come mostrato in figura 2.10.

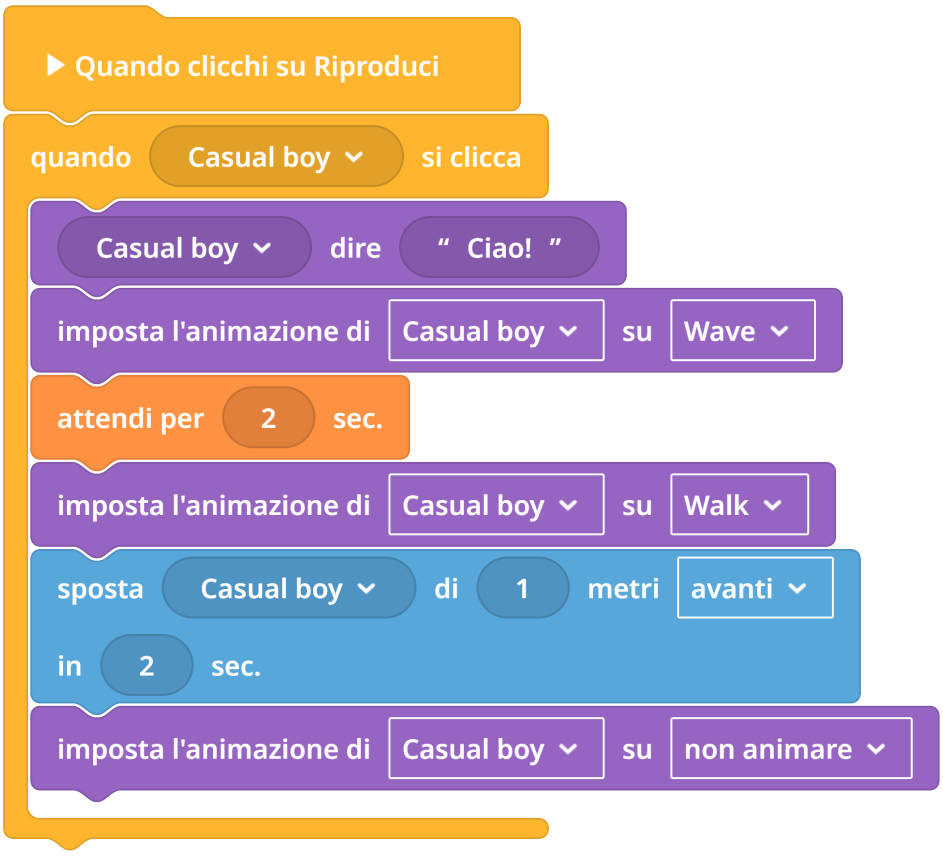



Figura 2.10: Esempio di script in CoSpaces Edu

Lo script proposto è alquanto banale: quando viene cliccato, l'oggetto "personaggio" (nell'immagine è l'oggetto "*Casual Boy*") dice "Ciao!", e viene attivata anche l'animazione "saluto". Dopo aver atteso 2 secondi, l'animazione cambia e, mentre il personaggio si muove, "cammina" in avanti; infine, la gestualità viene eliminata e il personaggio torna fermo. Questo esempio, molto semplice, è stato scelto per mostrare come sia possibile in CoSpaces programmare in maniera molto intuitiva e visuale, in un certo senso anche più "ludica" e concreta rispetto ad altri software di programmazione visuale. Infatti, in CoSpaces Edu, vi è un maggiore coinvolgimento emotivo dell'utente proprio grazie alla tridimensionalità e alla possibilità di interagire con gli oggetti e l'ambiente creato.

# Capitolo 3

# Sperimentazione e valutazione

In questo capitolo, si procederà con la descrizione della fase di sperimentazione, elemento centrale del lavoro di tesi, che ha coinvolto due classi di studenti presso il Liceo Classico Statale "Vincenzo Monti" di Cesena.

L'obiettivo primario di questa fase è stato valutare **in che modo l'adozione di micromondi come ambienti di apprendimento** - integrando *coding* e pensiero computazionale - **fosse fruttuoso in un contesto particolare come quello del Liceo Classico.**

Si è cercato di comprendere anche come tali concetti potessero essere integrati in modo significativo e formativo all'interno di un percorso didattico tradizionale, caratterizzato soprattutto da approcci umanistici. Inoltre, si è dedicata attenzione alla valutazione dell'efficacia e dell'utilità di Snap! e CoSpaces Edu (descritti nel Capitolo 2) come strumenti didattici per l'insegnamento della programmazione e la creazione di ambienti virtuali.

## 3.1 Contesto e percorso formativo intrapreso

La fase di sperimentazione si è svolta presso il **Liceo Classico Statale "Vincenzo Monti"** nella provincia di Forlì-Cesena.

Il Liceo Classico è un istituto di istruzione secondaria superiore che si caratterizza per un curriculum umanistico, focalizzato su materie quali latino, greco, storia, filosofia, letteratura italiana e storia dell'arte. Nonostante sia considerato prestigioso, spesso è circondato da pregiudizi che lo ritraggono come un percorso obsoleto e poco adatto al mondo del lavoro moderno, in quanto apparentemente carente di competenze pratiche.

In realtà, il Liceo Classico offre una formazione completa, capace di sviluppare competenze trasversali e pensiero critico, fungendo da solido trampolino per una vasta gamma di carriere professionali.

Tuttavia, va notato che il Liceo Classico non include materie legate all'informatica e alla programmazione. Pertanto, è risultato interessante e utile condurre un'indagine per valutare come l'insegnamento del *coding* e del pensiero computazionale possa essere integrato in modo efficace all'interno di un contesto come quello del Liceo Monti e quali benefici possa apportare agli studenti.

Nella creazione del percorso formativo, di fondamentale importanza è stato il contributo della Dott.ssa Ylenia Battistini, laureata in Ingegneria e Scienze Informatiche e componente del Comitato Scientifico di CRIAD Coding. Da anni segue e cura il progetto **Innova-Mente** [18][4], gestendo alcune iniziative e attività con le scuole del territorio cesenate: la sua esperienza e competenza hanno permesso di strutturare un percorso formativo che fosse non solo efficace, ma anche coinvolgente per gli studenti coinvolti.

Le due classi coinvolte nella fase sperimentale sono la 1Bc e la 2Bc, ognuna composta da 27 alunni. Per quanto riguarda la 1Bc, sono presenti 19 alunne e 8 alunni; nella classe 2Bc, invece, le ragazze sono 20 e i ragazzi 7. In tutto, dunque, hanno partecipato alla sperimentazione 54 alunni di cui il 72,2% erano donne e il restante 27,8% erano uomini. Tutti gli alunni hanno dai 14 ai 15 anni di età.

La sezione Bc presso il Liceo Monti è caratterizzata da una specifica attenzione alle materie scientifiche e offre un percorso di studi che prevede un'ora supplementare settimanale per approfondimenti in discipline come matematica, fisica, chimica e scienze naturali. Questa iniziativa si basa sulla consapevolezza che molti studenti, al termine del liceo, optano per proseguire gli studi in facoltà scientifiche, rendendo cruciale fornire una preparazione adeguata in tali materie.

La professoressa Manuela Piraccini, docente di matematica e informatica per le classi coinvolte, nota per la sua attenzione all'innovazione e alle nuove metodologie didattiche, ha manifestato interesse nell'introdurre la programmazione e il pensiero computazionale nel suo percorso didattico. Proprio lei ha proposto di coinvolgere le sue classi in un progetto pilota per valutare l'efficacia di tali competenze.

Nel progetto è stata coinvolta anche la professoressa Anna Ranieri, insegnante di latino, greco, storia e italiano, che ha mostrato interesse e ha suggerito di rendere il progetto interdisciplinare. L'obiettivo era consentire agli studenti di approfondire e consolidare le nozioni apprese in classe in un contesto più concreto e stimolante.

Con la collaborazione delle due insegnanti, è stato strutturato un percorso

[4] `https://www.innova-mente.org`

formativo incentrato sulla creazione di un micromondo virtuale dedicato alla matematica e alla storia delle civiltà antiche. La classe 1Bc ha sviluppato un micromondo ambientato nella Grecia antica, mentre la classe 2Bc ha focalizzato il suo progetto sull'Impero Romano. Queste ambientazioni sono state scelte in armonia con il programma di studio di storia, latino e greco, permettendo agli studenti di integrare le nozioni apprese in contesti più tangibili.

In definitiva, il progetto ha dimostrato di essere un valido strumento per unire competenze apprese in diverse materie, ancorandole in modo più duraturo nella memoria degli studenti.

La sperimentazione è durata complessivamente circa 15 ore per ciascuna classe, nei mesi di gennaio e febbraio 2024. Il percorso si è articolato in quattro fasi principali:

1. Introduzione al pensiero computazionale e alla programmazione. (2/3 ore)

2. Contestualizzazione e *brainstorming* per le idee da integrare nel micromondo. (1 ora)

3. Sviluppo (programmazione) del micromondo. (8/9 ore)

4. Presentazione dei lavori e valutazione finale. (2 ore)

**Fase 1** Poiché nessuno degli studenti coinvolti aveva esperienze pregresse di programmazione, si è dedicato un certo numero di ore all'introduzione dei concetti fondamentali del *coding*. In questo contesto, si è cercato di semplificare al massimo i concetti più tecnici del mondo della programmazione, e si è scelto di farlo utilizzando delle *metafore* [2]. Le metafore concettuali sono estremamente efficaci nelle prime fasi di apprendimento, in quanto offrono una spiegazione della nostra capacità di pensare e ragionare su concetti astratti: l'algoritmo diventa una ricetta, la memoria di un computer diventa una dispensa e le variabili diventano contenitori. In tal modo anche gli studenti meno esperti possono comprendere concetti complessi in modo più intuitivo.

Per passare alla parte pratica, si è scelto di utilizzare Snap! (Capitolo 2.1), uno strumento di programmazione particolarmente adatto a un pubblico giovane e inesperto.

Inizialmente, si è spiegato il concetto di algoritmo, mostrando come tale concetto fosse già presente nelle attività quotidiane degli studenti. Successivamente, si è proceduto a illustrare i concetti di base della programmazione, come variabili, cicli, condizioni e funzioni, evidenziando come tali concetti fossero già rappresentati in Snap! attraverso l'uso di blocchi.

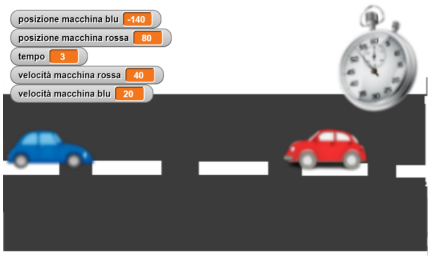
Come primo esercizio pratico, agli studenti è stato chiesto di creare un programma in grado di disegnare un quadrato (come illustrato nel Capitolo 2.1), sottolineando come ciò rappresentasse un esempio di algoritmo con un ciclo.

Come compito assegnato per casa, gli studenti hanno dovuto sviluppare uno script che disegnasse dei cerchi formando un "fiore" stilizzato. In questa fase, la sfida consisteva nel creare un codice il più breve ed efficiente possibile, e sono stati presentati diversi tentativi e soluzioni proposte dagli studenti. Di seguito si riporta la "soluzione" proposta per il compito a casa:


(a) Script (b) Blocco *custom*

Figura 3.1: Soluzione dell'esercizio sui cerchi assegnato per casa.

Come si può vedere in figura, lo script è davvero molto semplice, e ha richiesto l'utilizzo di pochissimi blocchi, uno dei quali è stato creato e personalizzato per creare il cerchio. Grazie a quest'esempio, i ragazzi hanno potuto apprezzare la possibilità di poter creare blocchi personalizzati, una funzionalità che permette un'ottima organizzazione del codice, che risulta essere molto più leggibile e manutenibile. Come ultimo esercizio, si è scelto di proporre una simulazione: calcolare il tempo (in secondi) di impatto tra due macchine che hanno velocità diverse.




In questo caso, è fondamentale il concetto di variabile: valori come le posizioni e le velocità dei due veicoli devono essere memorizzati dal sistema e il timer che cronometra il tempo di impatto deve essere aggiornato ad ogni ciclo. Inoltre, è stato necessario introdurre il concetto di condizione,

in quanto il programma deve essere in grado di riconoscere quando le due macchine si scontrano (e.g. hanno la stessa posizione) e di interrompere il ciclo.

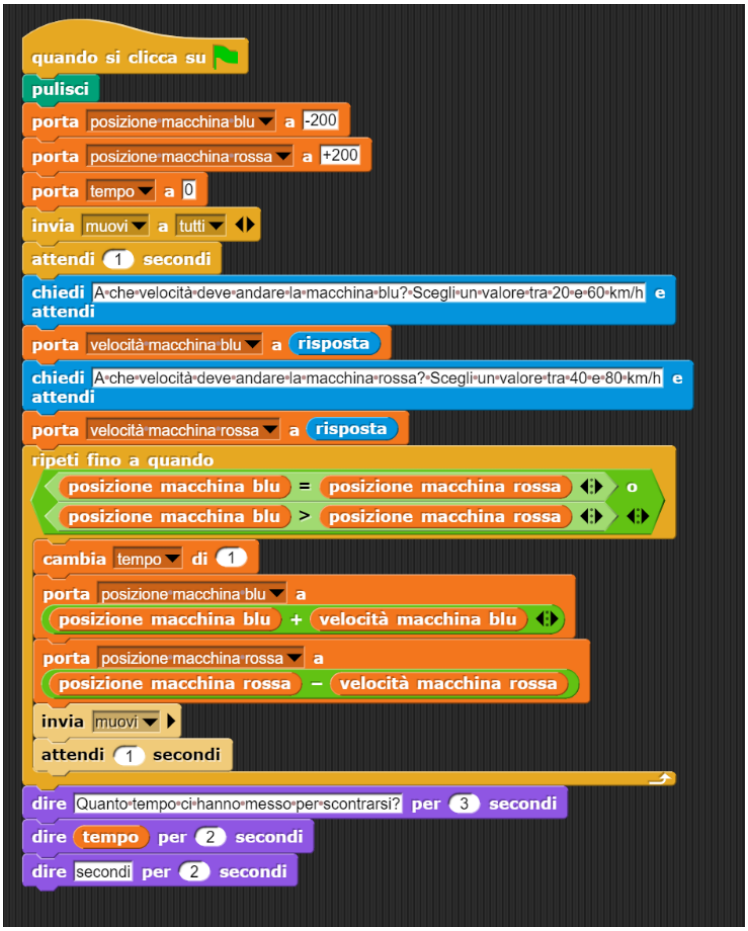



Figura 3.2: Script utilizzato per l'esercizio delle macchine.

**Fase 2** La seconda fase - molto breve - è stata dedicata al *brainstorming* e alla contestualizzazione del micromondo. In questa fase, gli studenti hanno avuto la possibilità di esprimere le proprie idee e di scegliere in che modo integrarle all'interno del micromondo. Inoltre, hanno creato piccoli gruppi di lavoro (in genere coppie o terzetti).

Si è scelto di fare anche una lezione introduttiva sulla Grecia antica (per la 1Bc) e l'Impero Romano (per la 2Bc), con particolare attenzione ai principali aspetti storici, culturali, geografici, letterari e matematici. La lezione, svolta insieme alle professoresse Piraccini e Ranieri, ha permesso di introdurre

i concetti chiave che sarebbero stati poi integrati nel micromondo virtuale, e ha permesso di stimolare la fantasia e la creatività degli studenti. Sempre in questo contesto, i ragazzi hanno scelto in che modo unire i lavori di tutti i gruppi per creare un unico micromondo.

Per quanto riguarda la classe 1Bc, ad esempio, si è scelto di partire da una scuola moderna: l'utente che accede al micromondo si trova in un corridoio scolastico, e può scegliere di entrare in quale aula vuole entrare. Ogni aula rappresenta un aspetto della Grecia antica, e l'utente può interagire con gli oggetti presenti per apprendere nuove nozioni; per tornare al presente, infine, deve superare delle sfide per testare le conoscenze acquisite. Gli argomenti trattati sono stati scelti da ogni gruppo, a seconda delle preferenze personali dei componenti. Tra di essi troviamo:

- Il moto parabolico, in particolare il lancio del giavellotto durante le battaglie narrate nell'Iliade.
- Il concetto di proporzione matematica nell'arte e il numero aureo.
- Pitagora e il suo teorema.
- Euclide e la geometria.
- La logica.
- Le macchine da guerra e la loro particolare costruzione.
- Diofanto di Alessandria.
- Talete e la sua filosofia.
- Il lancio del disco e le Olimpiadi antiche.
- La scuola di Atene, la scuola pitagorica e la scuola di Mileto.

Per quanto riguarda, invece, la classe 2Bc, si è scelto di creare un micromondo che permettesse all'utente di esplorare un'antica strada romana, in cui in ogni bottega è possibile teletrasportarsi in un ambiente diverso, dedicato ad uno tra gli argomenti scelti per il micromondo. Anche in questo caso, come per la 1Bc, gli argomenti trattati sono stati scelti dai ragazzi stessi.

- La crittografia e il cifrario di Cesare.
- La centuriazione e la costruzione di un campo romano.
- Gli anfiteatri antichi e i giochi gladiatorii.

- L'arte militare (armi, tattiche di guerra, ecc.).
- Il Colosseo e la sua costruzione.
- L'astronomia.
- La tassellazione e la costruzione di mosaici.
- Il calendario romano.
- La scultura e l'architettura romana.

Ciascun argomento, come si può intuire facilmente, si basa su principi matematici, ma anche storici, letterari e artistici, e ha permesso di concretizzare in modo più tangibile e coinvolgente le nozioni apprese in classe.

**Fase 3** La terza fase è stata la più importante, poiché è stata interamente dedicata alla programmazione del micromondo. Parte di questa fase è stata svolta in classe, con l'ausilio dei tablet resi disponibili dallo stesso Liceo, parte invece è stata svolta nel laboratorio di informatica, dove gli studenti hanno potuto lavorare con i computer. Per la programmazione del micromondo, si è scelto di utilizzare CoSpaces Edu (descritto in dettaglio nel capitolo 2.2). Questa scelta è stata fatta principalmente per volere della docente di informatica coinvolta, che ha preferito questo strumento per la creazione di ambienti virtuali più visivamente accattivanti e immersivi, grazie alla possibilità di visualizzarli in realtà virtuale. Inoltre, il liceo dispone sia di visori VR che dell'abbonamento "pro" del software, che offre una più vasta gamma di blocchi e funzionalità rispetto alla versione gratuita.

In questa fase, gli studenti hanno avuto la possibilità di mettere in pratica le nozioni apprese durante la fase di introduzione al *coding*, e di sperimentare in prima persona la creazione di un progetto più complesso e articolato. Questa fase è stata particolarmente formativa e stimolante, e ha permesso di mettere in luce le abilità e le competenze di ciascun studente, e di stimolare la creatività e l'originalità di ciascun gruppo. Inoltre, lavorare in piccoli gruppi ha permesso di sviluppare anche competenze trasversali, come il lavoro di squadra, la capacità di *problem solving* e la capacità di comunicare e collaborare con gli altri. Anche nel momento in cui si sono presentate delle difficoltà, gli studenti hanno potuto imparare a superarle in modo autonomo e a trovare soluzioni alternative, e questo ha permesso loro di crescere e di acquisire maggiore consapevolezza delle proprie capacità.

**Fase 4** L'ultima fase è stata dedicata alla presentazione dei lavori e alla valutazione finale. In questa fase, grazie all'utilizzo dei visori oculari, gli studenti hanno avuto la possibilità di entrare in prima persona nel micromondo che hanno creato, esplorandolo e imparando in modo leggero e divertente dai lavori dei propri compagni. I lavori sono stati valutati e si è voluto dedicare anche del tempo ad un momento di riflessione e di confronto, in cui gli studenti hanno potuto esprimere le proprie opinioni e valutare l'esperienza in modo critico e costruttivo. Durante questo momento, gli studenti hanno avuto la possibilità di esprimere le proprie opinioni in un *form* di valutazione, e di confrontare i due strumenti utilizzati, valutando l'efficacia e l'utilità di ciascuno di essi: una discussione più approfondita dei risultati di questa indagine verrà presentata nei prossimi paragrafi di questo capitolo.

## 3.2 Risultati ottenuti

**Questo percorso sperimentale si configura come un successo** sotto molteplici prospettive, poiché ha non solo raggiunto, ma superato gli obiettivi iniziali dei ragazzi, producendo risultati estremamente positivi.

I micromondi che hanno preso forma durante questa esperienza hanno davvero sorpreso anche gli studenti stessi: **ciascuno di loro è riuscito a raggiungere i propri obiettivi**, spesso senza neppure immaginare che le proprie competenze potessero tradursi in creazioni così straordinarie.

Sia gli studenti che le insegnanti hanno espresso un **entusiasmo e coinvolgimento** straordinari, testimoniando l'utilità e la formatività del progetto. L'esperienza è stata valutata come altamente benefica, non solo dal punto di vista didattico, ma anche da quello personale.

Gli studenti hanno dimostrato un **vivo interesse e partecipazione attiva**, evidenziando un apprendimento significativo e consolidando le competenze acquisite. Le insegnanti, d'altra parte, hanno notato una **crescita notevole nelle abilità e nelle capacità degli studenti**, confermando il valore aggiunto del percorso sperimentativo.

Il progetto ha contribuito non solo al progresso accademico dei discenti, ma ha anche favorito lo sviluppo di competenze personali e relazionali. La sinergia tra studenti e insegnanti ha creato un ambiente di apprendimento stimolante e collaborativo, dove l'entusiasmo ha alimentato la motivazione.

Complessivamente, questa esperienza ha dimostrato di essere non solo un successo pedagogico, ma anche un'opportunità formativa che ha arricchito il bagaglio di conoscenze e abilità di tutti i partecipanti, consolidando un legame positivo con l'apprendimento.

**MICROMONDI**

- 1Bc, "Dalla scuola antica alla scuola moderna: un viaggio nella matematica degli antichi Greci": `https://edu.cospaces.io/KBK-RYE`
- 2Bc, "Un giorno nella Roma antica: una passeggiata tra le magiche botteghe dell'Impero" : `https://edu.cospaces.io/ZNU-TNX`

Per quanto concerne la fase di raccolta dei feedback al termine del percorso, è stata adottato un approccio completo, coinvolgendo sia gli studenti che le insegnanti partecipanti attraverso un'indagine valutativa dettagliata. Quest'ultima è stata progettata insieme al Prof. Elvis Mazzoni, Professore Associato dell'Alma Mater Studiorum – Università di Bologna, specializzato nei processi educativi e creativi supportati dagli artefatti web di ultima generazione e dai robot[10]. Il suo contributo e le sue competenze in campo psico-pedagogico hanno permesso di strutturare un'indagine completa e approfondita, in grado di esplorare in modo dettagliato le percezioni e le valutazioni degli studenti e delle insegnanti coinvolte. In particolare, si vogliono citare due *papers* - (Mazzoni E., Benvenuti M.) che hanno ispirato il lavoro di sperimentazione ([9](2015) & [3](2020)) e la successiva fase di indagine.

Per quanto riguarda gli studenti, è stato implementato un *form* di valutazione mirato, progettato per esplorare aspetti ritenuti di particolare rilevanza. Inizialmente, è stato richiesto agli alunni di esprimere un giudizio complessivo sul lavoro svolto, con un focus specifico sull'efficacia e l'utilità del percorso, considerando anche l'implicazione delle materie coinvolte nel progetto.

Emergono con chiarezza i risultati positivi di questa fase dell'indagine, con tutti gli studenti che valutano in modo favorevole il progetto. La sua utilità e formazione sono stati riconosciuti universalmente, sottolineando l'apprezzamento per l'opportunità di mettersi alla prova in modo concreto e di farlo in un contesto di lavoro di gruppo, dove la negoziazione e la collaborazione hanno giocato un ruolo fondamentale.

Va notato, tuttavia, che alcuni studenti hanno indicato una difficoltà nel rispettare i tempi assegnati, considerandoli troppo stretti. Questo ha portato molti a continuare il lavoro sul progetto anche a casa. Nonostante questa criticità, l'entusiasmo generale rimane alto, e gli studenti si sono mostrati favorevoli a raccomandare il progetto ad altri colleghi, evidenziando la loro soddisfazione per il lavoro svolto e le valutazioni ricevute.

In sintesi, la raccolta dei feedback ha fornito un quadro dettagliato del successo del percorso sperimentativo, con l'apprezzamento degli studenti per

l'approccio pratico e collaborativo, nonostante le sfide temporali incontrate.

Un ulteriore aspetto approfondito durante l'indagine riguardava aspetti più ampi e trasversali dell'esperienza di apprendimento, andando oltre la stretta correlazione con il progetto. Gli studenti sono stati invitati a valutare autonomamente una serie di **competenze trasversali coinvolte**, tra cui la capacità di **risolvere problemi, il lavoro di squadra, la comunicazione, la collaborazione, la gestione del tempo e la creatività**. Oltre a questo, grazie alla loro esperienza diretta in classe, sono stati incoraggiati a descrivere come hanno affrontato le difficoltà e come hanno reagito di fronte agli errori, specialmente quando il codice non ha funzionato come previsto.

La lettura di queste risposte ha rivelato interessanti prospettive sulla gestione delle sfide e sugli approcci alla risoluzione dei problemi. È stato evidente come **i momenti di sconforto e stress siano stati affrontati in modo costruttivo dagli studenti**, spesso sostenuti dalla collaborazione e dal reciproco supporto. Emergono segnali positivi di crescita personale, con un netto **aumento del grado di autostima tra i ragazzi**. Inoltre, si nota una maggiore consapevolezza delle proprie capacità e dei propri limiti, suggerendo che l'esperienza ha non solo sviluppato competenze tecniche, ma anche un'intelligente gestione delle sfide e una prospettiva costruttiva nei confronti degli errori.

In questo contesto, il percorso formativo ha contribuito a plasmare non solo abilità specifiche legate al progetto, ma ha anche favorito lo sviluppo di competenze trasversali fondamentali per il successo personale e professionale degli studenti. La collaborazione e il supporto reciproco si sono dimostrati elementi chiave nel promuovere un ambiente di apprendimento positivo e nella formazione di individui consapevoli e resilienti.

L'ultima fase dell'indagine ha dedicato specifica attenzione all'efficacia e all'utilità dei due strumenti impiegati, in particolare Snap! e CoSpaces Edu. Con un focus particolare su CoSpaces Edu, intensivamente utilizzato durante lo sviluppo del micromondo, sono stati esplorati aspetti quali **l'usabilità, la facilità di apprendimento, la possibilità di personalizzazione e la capacità di lavorare in gruppo**. Ulteriori dettagli e riflessioni approfondite su questi risultati verranno presentati nel successivo paragrafo di questo capitolo.

Parallelamente, le insegnanti hanno partecipato a un'intervista conclusiva del percorso, esprimendo le proprie opinioni e valutazioni dall'angolazione di chi ha guidato e coordinato la parte didattica del progetto. Nonostante alcune difficoltà e criticità legate alla limitata disponibilità di tempo, anche in questo caso è emerso un giudizio complessivamente positivo. Questo riconoscimento della positività dell'esperienza sottolinea l'efficacia della conduzione didattica, nonostante le sfide incontrate.

La professoressa Piraccini (matematica e informatica) scrive: *"Il progetto della tesi sperimentale è stato particolarmente interessante e avvincente. Ha permesso ai ragazzi di lavorare in team, scoprire elementi nuovi e imparare a programmare e a sviluppare elementi di logica, a tirare fuori la loro creatività. Inoltre gli aspetti sviluppati sono certamente notevoli; ha stimolato nuove competenze e nuove soft skills. Elementi di criticità del progetto sono stati la mancanza di tempo per coordinare bene passo dopo passo i gruppi dei ragazzi, perché fra loro ci sono discenti che possono essere più preparati o meno predisposti. La classe risulta soddisfatta di quello che hanno fatto; sono orgogliosi di quanto sono riusciti a creare e ad apprendere. Per quanto riguarda l' utilità della didattica, la sua valenza può essere inserita solo in parte nella didattica tradizionale e in alcuni periodi prolungati nel tempo. Tali proposte possono essere realizzate distribuendole a contagocce perché diversamente non si riescono a svolgere gli elementi del programma dai inserire all'interno del contesto sperimentale nel modo dovuto. "*

La professoressa Ranieri (latino, greco, storia e italiano) scrive: *"Il progetto è stato davvero molto coinvolgente sia per docenti che per gli alunni. Sono stati più i punti di forza a caratterizzarlo che quelli di criticità . Per quanto riguarda i punti di forza, il progetto ha permesso a me docente di sperimentare una nuova modalità di lavoro, tanto da spingermi ad approfondire le mie conoscenze a riguardo con la partecipazione ad un corso di coding organizzato all'interno del mio istituto ("Le tecnologie nel contesto di ambienti e processi di apprendimento"). Gli alunni si sono rivelati i veri protagonisti dell'attività pensando in maniera creativa e stimolando la loro curiosità. Si sono messi alla prova cercando di risolvere problemi con l'applicazione della logica, ragionando passo dopo passo sulla strategia migliore per arrivare alla soluzione. Per quanto riguarda invece le criticità sottolineo che è stato a volte è stato difficile gestire le due classi coinvolte perché il tempo a disposizione, programmato per la realizzazione dell'attività, non è stato sempre sufficiente per esaurire tutte le problematiche e le richieste degli alunni. A ciò aggiungo soltanto quest'ultima osservazione: insegnando materie umanistiche ritengo che questa modalità di lavoro non possa essere sempre applicata alla mia didattica, ma soltanto a determinati aspetti, dopo un mirato lavoro di preparazione. Detto ciò ritengo comunque che l'esperienza sia stata molto valida, in quanto fonte di arricchimento per il corpo docente e per gli alunni."*

In conclusione, questo percorso sperimentale si rivela un successo sotto molteplici aspetti, con risultati positivi che emergono dalle diverse fasi dell'indagine condotta. Gli studenti hanno espresso un forte entusiasmo e coinvolgimento, apprezzando l'opportunità di mettersi alla prova concretamente in un conte-

sto collaborativo. La valutazione positiva del progetto si estende anche alla crescita personale degli studenti, evidenziando un aumento del grado di autostima e una maggiore consapevolezza delle proprie competenze. Nonostante alcune sfide e criticità, anche il giudizio positivo delle insegnanti conferma il successo complessivo del percorso, sottolineando il valore formativo e l'efficacia della conduzione didattica. In definitiva, il progetto ha non solo raggiunto gli obiettivi prefissati ma ha anche contribuito in modo significativo allo sviluppo sia delle competenze specifiche sia delle abilità trasversali degli studenti, consolidando un'esperienza formativa di successo.

## 3.3 Analisi critica degli strumenti utilizzati

In questa sezione si intende analizzare in modo critico i due strumenti utilizzati durante il percorso sperimentativo, Snap! e CoSpaces Edu, con l'obiettivo di mettere in luce le principali differenze e criticità emerse durante l'esperienza d'uso. Il primo si è rivelato molto utile per introdurre i concetti base della programmazione e per prendere familiarità con la logica a blocchi, una fase fondamentale in quanto quasi nessuno degli studenti aveva mai avuto esperienze pregresse di *coding*. Il secondo strumento, invece, è stato introdotto per volere della docente di informatica coinvolta, poiché preferito per la creazione di applicazioni più visivamente accattivanti e immersive, grazie alla possibilità di visualizzarli in realtà virtuale.

Come si evince dalle descrizioni dei due strumenti (capitolo 2), Snap! e CoSpaces Edu presentano molte somiglianze, ma si ritiene necessario anche mettere in luce alcune differenze significative e criticità emerse durante l'esperienza d'uso.

In qualità di *tools* progettati per fornire un supporto all'apprendimento, entrambi i software si sono rivelati piuttosto efficaci e particolarmente adatti a un pubblico giovane e inesperto. Entrambi offrono un'interfaccia utente molto intuitiva e visivamente accattivante, che rende l'esperienza complessiva piacevole e sicuramente più coinvolgente rispetto alla didattica tradizionale. Si apprezza in modo particolare anche la possibilità di mettere in risalto una lunga serie di competenze e nozioni trasversali per la creazione di lavori interdisciplinari, uno sforzo che si ritiene molto apprezzabile nella didattica moderna.

Tuttavia, in base a quanto emerso nell'indagine condotta alla fine della sperimentazione, gli studenti hanno riscontrato alcune difficoltà riguardanti l'utilizzo di CoSpaces Edu. Nonostante fosse stata messa a disposizione la versione "pro" del software, che offre un'ampia gamma di blocchi e funzionalità, gli studenti hanno individuato delle mancanze e delle criticità che li hanno portati a dover integrare l'utilizzo di CoSpaces Edu con altri strumenti.

Tra questi, ad esempio, si ricorda l'utilizzo di GeoGebra [5] per la creazione di grafici matematici e di ThingLink [6] per la creazione di immagini interattive. CoSpaces Edu, infatti, non possiede nessuno strumento interno di disegno e modellazione 3D, e la sua libreria, per quanto vasta, non è sufficiente a coprire tutte le esigenze degli studenti. A riguardo di ciò, infatti, i ragazzi hanno lamentato la marcata similarità tra i lavori dei diversi gruppi e la difficoltà di personalizzare e rendere unici i propri progetti. Questa problematica non è stata, invece, riscontrata in Snap!, dove gli studenti hanno potuto disegnare e personalizzare i propri personaggi e sfondi, e creare progetti molto più originali e diversificati. In alcuni casi, per quanto riguarda il micromondo finale in 3D, si è addirittura scelto di ricercare online (principalmente su Sketchfab [7]) modelli già pronti e di importarli in CoSpaces Edu per arricchire il proprio progetto. Inoltre, su CoSpaces, l'organizzazione del codice risulta essere significativamente meno efficiente rispetto a Snap!. In primo luogo, non vi è alcun modo di costruire dei blocchi personalizzati, ma soltanto delle funzioni, dovendo dunque ricorrere ad un concetto non proprio intuitivo per gli studenti. In aggiunta, le funzioni create sono utilizzabili soltanto nella scena in cui vengono create. In secondo luogo, non è presente - almeno per ora - alcuna funzionalità che permetta di raccogliere dati in input dall'utente. Questa *feature* sarebbe stata particolarmente utile nella creazione di esercizi e domande interattive e raccogliere le risposte dell'utente. Oltre a ciò, per creare sfide sempre nuove, si è stati costretti ad utilizzare blocchi per la randomizzazione di valori numerici (e.g. i dati di un problema di matematica), ma non è stato possibile creare un vero e proprio sistema di generazione casuale di domande e risposte. I quiz proposti, infatti, risultavano sempre simili e prevedibili, e non hanno stimolato particolarmente gli studenti. Infine, è stata segnalata una considerevole difficoltà nel lavorare sullo stesso progetto contemporaneamente, in quanto non vi è alcuna funzionalità di *versioning* condivisa e di *collaboration* in tempo reale.

Un punto a favore di CoSpaces Edu, invece, è stata la possibilità di "entrare" in prima persona nel micromondo che le classi hanno creato e assemblato. Grazie all'utilizzo dei visori forniti dallo stesso Liceo, infatti, i ragazzi hanno condiviso un momento di grande entusiasmo e coinvolgimento, e hanno potuto apprezzare il lavoro svolto in modo molto più concreto e tangibile. Questo tipo di "avventura" non sarebbe stata possibile con Snap!, che, come già detto, lavora con ambienti bidimensionali. Muoversi all'interno dell'ambiente, interagire direttamente con gli oggetti e affrontare le sfide proposte ha reso possibile la creazione di un'esperienza di apprendimento senza precedenti, dove l'utente impara divertendosi e, in questo caso, anche partecipando attivamente alla

[5] `https://www.geogebra.org/`
[6] `https://www.thinglink.com/`
[7] `https://sketchfab.com/`

costruzione del micromondo.

In conclusione, l'utilizzo di entrambi i software si è rivelato molto utile e formativo, e ha permesso di raggiungere gli obiettivi prefissati. Dal punto di vista delle insegnanti, CoSpaces Edu è stato ritenuto efficace anche se eccessivamente costoso e non all'altezza delle aspettative. Snap!, invece, è stato apprezzato per la sua semplicità e per il fatto che stimoli in modo particolare la creatività e l'originalità degli studenti.

# Capitolo 4

# Idee per nuove piattaforme per la creazione di micromondi digitali

A valle di delle attività sperimentali svolte, è possibile affermare che i **micromondi digitali rappresentano un'ottima soluzione educativa**, valida anche in caso di insegnamenti complessi e progetti interdisciplinari. Tuttavia, è importante sottolineare che la creazione di un micromondo digitale non è un'operazione banale, ma richiede un'attenta pianificazione e un adeguato supporto tecnologico.

## 4.1 Riflessione su sviluppi futuri e prospettive per una piattaforma educativa integrata"

Partendo dalle considerazioni emerse nel Capitolo 3, in cui si elaborano i feedback raccolti dagli studenti e dagli insegnanti coinvolti, si è deciso di approfondire la ricerca di **soluzioni alternative per la creazione di micromondi digitali**, al fine di individuare strumenti più adatti e più performanti rispetto a quelli utilizzati nella fase sperimentale.

Tuttavia, non è semplice individuare una soluzione che soddisfi tutte le esigenze d'uso: per di più, **l'integrazione omogenea di strumenti già esistenti è un'operazione estremamente complessa, e non sempre realizzabile**. Questo accade soprattutto quando si lavora con strumenti proprietari, che non permettono l'accesso al codice sorgente o che non offrono la possibilità di estendere le funzionalità esistenti: un esempio è proprio CoSpaces (Capitolo 2.2).

Una prima strada da percorrere potrebbe essere quella di sviluppare un'estensione per Snap!, che, come abbiamo visto, si configura come uno strumento molto flessibile e adattabile. La questione principale, in questo caso, è quella di capire come aumentare l'immersività e l'interattività di un sistema bidimensionale come quello che manda in esecuzione Snap!. Si potrebbe fare progetti migliorativi, facendo riferimento a JavaScript e a librerie come Three.js, che permettono di creare e gestire ambienti tridimensionali.

Idealmente, la soluzione migliore - anche se complessa e ambiziosa - sarebbe quella di sviluppare un nuovo strumento che integri le funzionalità più utili e interessanti sia degli strumenti di **programmazione visuale** utilizzati per questa tesi, sia di tecnologie per la **realtà virtuale**, la **collaboratività in tempo reale** e la ***gamification***.

In primo luogo, l'obiettivo principale sarebbe quello di creare un ambiente gratuito, ma soprattutto ***open source***, in modo da promettere la massima trasparenza e la massima flessibilità. A tal proposito, si ritiene che una soluzione basata sul **web** sia la migliore, in quanto permette di raggiungere un pubblico più ampio e di garantire la massima accessibilità.

L'elemento fondamentale di questa soluzione sarebbe la programmazione visuale, che permette di rendere l'esperienza di programmazione più accessibile e più divertente, soprattutto per gli studenti più inesperti. La logica dei blocchi deve essere chiara, intuitiva, deve permettere di creare programmi complessi ma lasciare spazio anche alla creazione di blocchi personalizzati. Come per Snap!, si ritiene che il linguaggio più adatto sia JavaScript (anche se non è l'unica opzione valida), in quanto permette di creare programmi complessi e di interfacciarsi con numerosi *framework* e librerie esterne.

Uno step successivo sarebbe quello di integrare funzionalità per fare in modo che gli utenti cooperino in tempo reale, sia nella creazione dei progetti, sia nella loro fruizione. Un esempio di questa funzionalità si ritrova in Croquet [8]. Si tratta di un ambiente di sviluppo *open source* che permette di creare applicazioni collaborative in tempo reale, con una tecnologia *serverless*. Anche Croquet si basa su JavaScript (in particolare, su Node.js) e permette di creare e collaborare a distanza in ambienti 3D e di gestire la realtà virtuale.

Quest'ultima, idealmente, costituisce il passo successivo verso la creazione di un micromondo digitale più completo: come visto, la realtà virtuale permette di sperimentare un apprendimento drasticamente più coinvolgente e interattivo. La realtà virtuale sul web è una tecnologia in rapida evoluzione, ma al contempo ampiamente consolidata; nell'ambito di questa tesi, in particolare, si sono approfondite la libreria Three.js [9] e WebXr, una specifica API

[8] `https://croquet.io/index.html`
[9] `https://threejs.org/`

Web per la VR [10].

Three.js è una libreria JavaScript che semplifica la creazione di grafica 3D interattiva su pagine web. Fornisce un'interfaccia facile da usare per creare scene 3D, oggetti, luci, materiali e animazioni. Ideale per la creazione di esperienze 3D su pagine web, giochi, visualizzazioni dati 3D e applicazioni simili. Può essere utilizzata in combinazione con HTML5, CSS3 e WebGL. Three.js gestisce il *rendering* di scene 3D attraverso WebGL o, se non disponibile, utilizza automaticamente alternative come SVG o HTML5 Canvas.

WebXR è una specifica API Web che consente di creare esperienze di realtà aumentata (AR) e realtà virtuale (VR) su pagine web. Fornisce una serie di API che consentono ai browser di interagire con dispositivi di realtà virtuale e aumentata: supporta diverse tecnologie, tra cui i visori oculari. WebXR offre funzionalità per la gestione di visori e dispositivi AR, controllandone il posizionamento e il tracciamento del movimento, oltre a una serie di funzionalità specifiche per l'interazione in ambienti 3D.

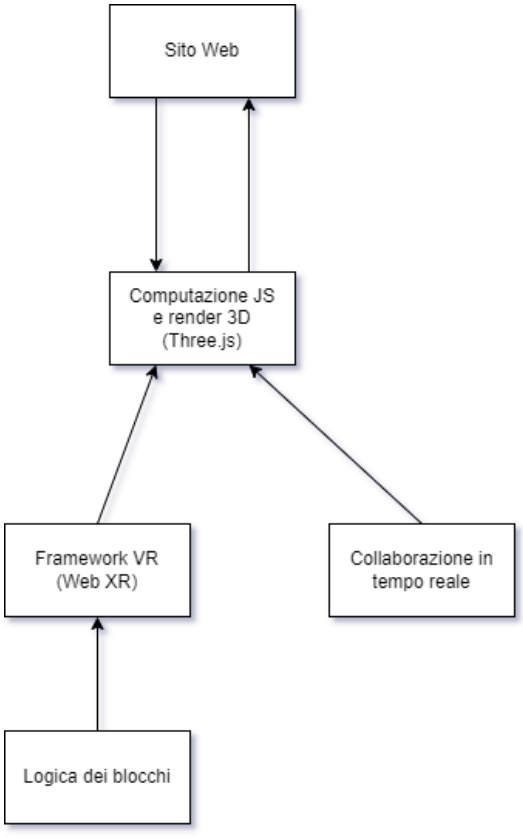



Figura 4.1: Idea di architettura.

[10] `https://immersiveweb.dev`

In Figura 4.1 è possibile osservare una proposta architetturale di una soluzione integrata per uno strumento educativo digitale completo. Si tratta di un'idea iniziale, che prevede la creazione di un ambiente di programmazione visuale basato su JavaScript, che permette di creare progetti collaborativi in tempo reale e di interfacciarsi con tecnologie per la realtà virtuale. La pagina web, in particolare, è sostenuta da moduli di computazione JavaScript, che permettono sia di gestire la logica dei blocchi, sia di renderizzare ambienti 3D e di gestire la realtà virtuale. Il tutto è sostenuto da un server Node.js, che permette di gestire la collaborazione in tempo reale e di gestire la comunicazione tra i vari utenti.

In ultimo, si potrebbe prevedere la possibilità di interfacciarsi con l'ambiente esterno, ad esempio attraverso l'***Internet of Things***, utilizzando sistemi *embedded* come Arduino o Raspberry. Questo tipo di tecnologia permetterebbe ai ragazzi di sviluppare una maggiore consapevolezza dell'ambiente circostante e di come esso possa essere influenzato e controllato attraverso la programmazione.

## 4.2 Verso un Eduverso: ambizioni, principi e sfide delle tecnologie educative

Entrambe le prospettive proposte sono estremamente ambiziose e richiedono un impegno notevole, ma potrebbero portare a risultati di grande impatto e innovazione.

Il filo conduttore, però, deve essere lo stesso, qualunque sia la strada che si vuole intraprendere: i percorsi di apprendimento proposti devono necessariamente essere *student-centered*. Le nuove tecnologie devono contribuire a cambiare, arricchire e migliorare l'esperienza di apprendimento. Come affermato da Cicconi e Marchese[19], questi ambienti devono avere le seguenti caratteristiche non negoziabili:

- Devono **facilitare l'apprendimento** del contenuto visivo.
- Devono permettere agli studenti di **imparare facendo** (*learn by doing*), assumendo un ruolo attivo nel processo di apprendimento.
- Devono aumentare l'*engagement* e la **motivazione dei discenti**, fattore molto importante nella buona riuscita del percorso di apprendimento.
- Devono **favorire la collaborazione e l'interazione** sociale.

Si ritiene importante anche tenere in considerazione anche gli studenti con disabilità o con Bisogni Educativi Speciali (BES), che potrebbero trarre grandi

benefici da un ambiente di apprendimento digitale. Le nuove tecnologie possono e devono essere considerate come un **supporto valido per l'inclusione e l'accessibilità**, e non come un ostacolo. Infine, bisogna sempre considerare il benessere degli studenti, e garantire che l'uso delle nuove tecnologie non diventi un'ulteriore fonte di stress e ansia, ma un'opportunità di crescita e arricchimento.

In questo senso, il focus deve essere sempre sulla creazione di nuovi scenari *onlife* [6](Floridi, 2019) di valore. Anche il Ministero dell'Istruzione e del Merito (MIM), nel 2022, ha delineato il "Piano Scuola 4.0". In questo documento viene messo in evidenza come il multiverso in ambito educativo, ovvero l'**eduverso**[11], "possa offrire la possibilità di ottenere nuovi "spazi" di comunicazione sociale, maggiore libertà di creare e condividere, offerta di nuove esperienze didattiche immersive attraverso la virtualizzazione, creando un continuum educativo e scolastico fra lo spazio fisico e lo spazio virtuale per l'apprendimento, ovvero un ambiente di apprendimento *onlife*".

Al momento, il concetto di **metaverso** non può dirsi ancora pienamente realizzato, ma è possibile intravedere i primi passi in questa direzione, soprattutto grazie a tecnologie come la **realtà virtuale e la realtà aumentata**. Citando Mystakidis (2022) [12], si può dire che la realtà virtuale in particolare sia oggi una possibilità concreta per l'implementazione di attività in ambito educativo.

Mancano, però, dei punti fondamentali che, solo se presenti contemporaneamente, permetterebbero di definire questa condizione come "eduverso": innanzitutto, la mancanza di realismo, e dunque la sensazione di essere psicologicamente ed emotivamente immersi nella realtà alternativa; in secondo luogo, l'ubiquità, la facoltà di accedere a questo mondo virtuale da qualsiasi *device* e in qualsiasi momento. Si ha poi l'interoperabilità, ovvero la possibilità di interagire con altri utenti e con il mondo virtuale in modo naturale (attraverso degli *avatar* e *asset* digitali) e senza limiti; infine, la scalabilità, in modo che la tecnologia riesca a tenersi al passo con le prestazioni elevate richieste da questo tipo di sistemi.

In generale, vari studi sull'applicazione della realtà virtuale in ambito educativo ne evidenziano una grande potenzialità, specialmente per l'implementazione di un **senso di presenza** (Krassmann, 2019)[7], fondamentale anche in questo periodo post-pandemico, in cui la didattica a distanza ha messo in evidenza la necessità di un contatto più diretto e coinvolgente con gli studenti.

# Capitolo 5

# Conclusioni

L'obiettivo primario di questa tesi è stato quello di verificare l'efficacia del coding e del pensiero computazionale come strumenti per una didattica innovativa, in particolare in un contesto scolastico basato sull'approfondimento di discipline umanistiche.

Il punto di partenza del percorso di sperimentazione vedeva degli alunni coinvolti - studenti del Liceo Classico "Vincenzo Monti" di Cesena - non particolarmente avvezzi all'uso di strumenti informatici e digitali. Nello specifico, nessuno di loro aveva mai avuto esperienze di programmazione.

Si è cercato di valutare la proficuità di un **approccio ludico e interattivo**, basato sulla creazione di micromondi digitali, per introdurre i concetti di base della programmazione e del pensiero computazionale, al fine di sviluppare una serie di competenze trasversali e di promuovere l'acquisizione di conoscenze in un contesto più leggero rispetto a quello tradizionale.

Per fare ciò, si è scelto di utilizzare due strumenti di programmazione visuale, Snap! e CoSpaces Edu. Il primo è stato utilizzato per introdurre i concetti base della programmazione, e si è rivelato particolarmente efficace quando associato a metafore concettuali e a esercizi di *problem solving*. Il secondo, invece, è stato utilizzato per creare micromondi digitali più complessi e coinvolgenti, in quanto offre la possibilità di creare ambienti 3D e di interagire con essi attraverso la realtà virtuale.

I risultati ottenuti sono stati molto positivi: gli studenti coinvolti si sono dimostrati particolarmente interessati e coinvolti, e hanno dimostrato di aver acquisito una serie di competenze trasversali, come la capacità di collaborare in gruppo, la capacità di *problem solving* e la capacità di pensiero critico. Anche le insegnanti coinvolte hanno espresso un giudizio positivo, sottolineando come l'uso di questi strumenti abbia permesso di uscire dalla routine e di introdurre un approccio più interdisciplinare e più coinvolgente.

Tuttavia, è importante sottolineare che la creazione di un micromondo

digitale non è un'operazione banale, ma richiede un'attenta pianificazione e un adeguato supporto tecnologico. In particolare, i due strumenti utilizzati nella fase sperimentale, Snap! e CoSpaces Edu, hanno mostrato alcune limitazioni e difficoltà di utilizzo, soprattutto per quanto riguarda la collaborazione in tempo reale e la scarsa flessibilità nella gestione di alcune funzionalità.

A valle di queste considerazioni, si è deciso di approfondire la ricerca di soluzioni alternative per la creazione di micromondi digitali, al fine di individuare strumenti più adatti e più performanti rispetto a quelli utilizzati nella fase sperimentale. In particolare, si è proposta una soluzione integrata per uno strumento educativo digitale completo, che permetta di creare progetti collaborativi in tempo reale e di interfacciarsi con tecnologie per la realtà virtuale.

Da queste riflessioni emerge un quadro tecnologico sicuramente avanzato, al passo con le diverse tecnologie emergenti e con le esigenze di un mondo sempre più interconnesso e sempre più digitale. Tuttavia, si registra anche una situazione in cui gli strumenti esistenti non sempre permettono un'implementazione sostenibile e flessibile. Persino quello *open source* non è un mondo integrato, e spesso si trova a dover fare i conti con limitazioni e difficoltà di utilizzo da parte di utenti memo esperti.

In ogni caso, si ritiene che il modo migliore per procedere e affrontare un progetto come quello proposto sia quello di essere affiancati da persone competenti nelle discipline informatiche, che possano supportare e guidare il processo di sviluppo e di implementazione. A fianco di queste figure, è importante anche la presenza di figure competenti nelle discipline didattiche coinvolte; soltanto il sinergico lavoro di queste due parti può portare a una buona riuscita di un progetto di questo tipo.

Guardando al futuro, è innegabile che la direzione intrapresa rappresenti un passo significativo, meritando ulteriori investimenti e attenzioni. Un punto cruciale risiede nella formazione degli insegnanti, essenziale per garantire una gestione competente di queste tecnologie e una valutazione accurata dei requisiti iniziali e dei risultati conseguiti.

È molto importanza proseguire gli sforzi nella ricerca e nello sviluppo di strumenti più avanzati, superando le attuali soluzioni per assicurare un'esperienza d'uso più appagante e una maggiore efficacia didattica. In questo contesto, investire nelle competenze docenti è una strategia chiave per sfruttare appieno il potenziale delle nuove tecnologie in ambito educativo.

Un aspetto notevole è l'importanza che il mondo dell'educazione sta finalmente riconoscendo nei confronti delle nuove tecnologie, cercando di integrarle in maniera più organica ed efficace. Auspichiamo un continuo aumento dell'attenzione su queste tematiche, con la speranza di vedere progetti simili sempre più diffusi e integrati in ambienti scolastici e formativi.

**È cruciale che la scuola diventi un luogo in cui l'apprendimento non sia fonte di stress o noia, ma un'esperienza coinvolgente e stimolante, se non persino divertente**. Condividere l'idea di Quintiliano di unire gioco e apprendimento emerge come una strategia eccellente per rendere la formazione più leggera ed efficace, contribuendo a garantire che gli studenti crescano come individui consapevoli di sé stessi e degli altri.

# Ringraziamenti

Innanzitutto, desidero esprimere la mia sincera gratitudine al Professor Alessandro Ricci, il mio relatore, per la sua guida competente e il suo prezioso sostegno durante tutto il percorso di elaborazione di questa tesi. Senza le sue mille idee e la sua infinita esperienza, questo lavoro non sarebbe stato possibile.

Voglio ringraziare di cuore il Professor Elvis Mazzoni: la sua infinita gentilezza è stata un punto di riferimento durante tutto il percorso di sperimentazione.

Un ringraziamento particolare va alla Dott.ssa Ylenia Battistini, una colonna portante di questo lavoro: i suoi preziosi consigli e la sua esperienza sono stati fondamentali per meno.

Ringrazio anche le professoresse Manuela Piraccini e Anna Ranieri, e i ragazzi della 1Bc e della 2Bc: senza il vostro supporto e il vostro prezioso contributo questo lavoro non sarebbe mai stato possibile. Vi porterò per sempre nel cuore.

Infinitamente grazie alla mia famiglia: grazie Mamma, Papà e Ale per il sostegno e l'incoraggiamento che mi avete dato in questi anni, anche quando era più difficile starmi accanto. Non avrei mai raggiunto questo traguardo senza di voi.

Lorenzo, ti sono grata per il tuo amore e il tuo supporto, per avermi sopportato nei momenti di stress e per avermi sempre incoraggiato a dare il meglio di me stessa.

Un grazie speciale va a Simo e Matteo, persone speciali e fondamentali nella mia quotidianità, sempre presenti e pronti a strapparmi un sorriso.

Sono grata a tutti i miei amici, in particolare ai ragazzi conosciuti durante il mio percorso universitario: grazie per aver reso questi anni indimenticabili. Grazie Ale, Dani, Denny, Ema, Jack, Marco, Paso, Simo, Roby e Zava: le risate, i poker, le serate cinema e le chiacchierate in poli hanno reso questi anni i più belli della mia vita.

Grazie anche ad Andrea Silenzi e Daniele Arcangeli e a tutta la squadra del Budokan: siete un punto di riferimento e per me e in questi anni mi avete trasmesso tanti valori che mi porto nel cuore.

Grazie a tutti, per esserci stati nei momenti in cui tutto sembrava impossibile e in quelli più spensierati.

# Bibliografia


[1] Lister R. Ahadi, A. Geek genes, prior knowledge, stumbling points and learning edge momentum: parts of the one elephant? *Proceedings of the ninth annual international ACM conference on International computing education research (ICER '13)*, 2013.

[2] Charoula Angeli and Michail Giannakos. Computational thinking education: Issues and challenges. *Computers in Human Behavior*, 105:106185, 2020.

[3] Martina Benvenuti and Elvis Mazzoni. Enhancing wayfinding in preschool children through robot and socio-cognitive conflict. *British Journal of Educational Technology*, 51(2):436–458, 2020.

[4] Jerome Bruner. *The Process of Education*. Harvard Univer. Press., 1960.

[5] Various contributors. *Vygotsky and Education: Istructional implications and Applications of socio-historical psychology*. Cambridge University Press, 1990.

[6] L. Floridi. What the near future of artificial intelligence could be. *Philos. Technol.*, 2019.

[7] Melo M. Pinto D. Peixoto B. Bessa M. Bercht M. Krassman, A.L. What is the relationship between the sense of presence and learning in virtual reality? *A 24-Year Systematic Literature review.*, 2019.

[8] Shuchi L Mason and Peter J Rich. Preparing elementary school teachers to teach computing, coding, and computational thinking. *Contemporary Issues in Technology and Teacher Education*, 19(4):790–824, 2019.

[9] Elvis Mazzoni and Martina Benvenuti. A robot-partner for preschool children learning english using socio-cognitive conflict. *Journal of Educational Technology  Society*, 18(4):474–485, 2015.

[10] Benvenuti M. Mazzoni E. *Developmental technologies: Evoluzione tecnologica e sviluppo umano*. Maggioli Editore, 2019.

[11] Elena Mosa, Andrea Benassi, and Silvia Panzavolta. Dibattere e argomentare in realtà virtuale: primi esiti di una sperimentazione sul campo. *IUL Research*, 4(7):70–88, 2023.

[12] S. Mystakidis. Metaverse. *Encyclopedia*, 2022.

[13] Seymour Papert. Teaching children thinking. *Journal of Structural Language*, 1975.

[14] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books Inc., New York, 1980.

[15] Seymour Papert. *Constructionism: A New Opportunity for Elementary Science Education - A proposal to the National Science Foundation*. MIT, Cambridge, 1989.

[16] Seymour Papert. *Children's Machine: Rethinking school in the age of the computer*. 1993.

[17] Mitchel Resnick. *Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play*. The MIT Press, 2018.

[18] Alessandro Ricci, editor. *Quaderni Innova-Mente: Micromondi a Scuola*. v. 0.9-20221109 edition, 2022. Con contributi di tutto il gruppo Innova-Mente.

[19] M. Marchese S. Cicconi. Augmented learning: an e-learning environment in augmented reality for older adults. *INTED2019 proceedings*, 2019.

[20] Anuchart Threekunprapa and Prachyanun Yasri. Unplugged coding using flowblocks for promoting computational thinking and programming among secondary school students. *International Journal of Instruction*, 13(3):207–222, 2020.

[21] Jeannette M. Wing. Computational thinking. *Commun. ACM*, 49(3):33–35, mar 2006.

[22] Jeannette M. Wing. Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology*, 25(2):7–14, 2017.