UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Laurea Magistrale in
Informatica per la Comunicazione

PROGETTAZIONE E SVILUPPO DI UN PERCORSO DIDATTICO SULLA RICORSIONE PER IL PROBLEM SOLVING E DI UN'INFRASTRUTTURA SOFTWARE DI SUPPORTO

RELATORE
Prof.ssa Violetta Lonati

TESI DI LAUREA DI
Manuel Previtali
Matr. 844310

CORRELATORI
Prof. Dario Malchiodi
Prof.ssa Anna Morpurgo

Anno Accademico 2014/2015

Indice

Introduzione

Il lavoro descritto in questa tesi è stato svolto presso ALaDDIn –Laboratorio di Didattica e Divulgazione dell'Informatica– del Dipartimento di Informatica dell'Università degli Studi di Milano. L'obiettivo principale di questo lavoro è stata la progettazione di un percorso didattico, rivolto alle scuole secondarie di primo grado, che affronta il tema della ricorsione come tecnica per il *problem solving*. *Problem solving* è un'espressione che in ambito informatico si riferisce alla capacità di trovare soluzione ai problemi mediante la progettazione e l'applicazione di algoritmi. Il *problem solving* è un processo creativo, che richiede a sua volta pensiero computazionale e capacità di astrazione e può dunque avere un importante valore formativo fin dalla scuola primaria e secondaria di primo grado.

Insegnando questa tecnica a studenti di giovane età, ci si aspetta che essi possano ampliare i propri orizzonti cognitivi. Per l'insegnamento di questo tema si è progettato un percorso didattico suddiviso in quattro fasi che mirano al raggiungimento graduale di specifici obiettivi formativi da parte degli studenti partecipanti. Nelle fasi previste dal percorso gli studenti dovranno analizzare alcuni algoritmi attraverso una modalità motoria e manipolatoria (che viene definita *algomotoria*) e utilizzando uno strumento software appositamente progettato e sviluppato; l'utilizzo da parte degli studenti dello strumento software è inoltre guidato da una scheda di lavoro. Il percorso si conclude con la progettazione di un semplice algoritmo ricorsivo.

È stata inoltre effettuata una sperimentazione del percorso e dei relativi materiali (il software e la scheda), lavorando in una classe terza di una scuola secondaria di primo grado di Milano.

Ci si è inoltre dedicati alla progettazione e allo sviluppo di un'infrastruttura software basata sulla piattaforma *Docker*, con lo scopo di supportare l'utilizzo dello strumento software appena descritto. La progettazione dell'infrastruttura è avvenuta nell'ottica di integrarvi anche gli strumenti software usati in altri percorsi didattici proposti da ALaDDIn. L'infrastruttura inoltre consente di tracciare e archiviare in un database *MongoDB* alcune informazioni legate all'utilizzo dei software didattici integrati, al fine di poter

effettuare delle eventuali analisi statistiche sull'efficacia dei percorsi didattici. Dopo aver progettato l'infrastruttura, sono stati integrati due strumenti software per verificare che l'infrastruttura avesse la flessibilità auspicata.

La tesi presenta la struttura seguente. La prima parte è dedicata alla ricorsione come tecnica per il *problem solving* ed è suddivisa in diversi capitoli. Il capitolo 1 richiama i concetti base relativi al tema della ricorsione, quindi propone una rassegna degli approcci didattici che affrontano questo tema. Il capitolo 2 descrive i presupposti su cui ci si è basati per la progettazione del percorso didattico, delineandone gli obiettivi formativi e descrivendo il contesto e la metodologia che lo caratterizzano. Il capitolo 3 presenta le diverse fasi che compongono il percorso, descrivendone le attività previste e i materiali proposti, evidenziando alcune considerazioni didattiche e approfondendo in particolare gli aspetti relativi alla progettazione del software e della scheda di lavoro che lo accompagna. Il capitolo infine riassume gli esiti della sperimentazione del percorso didattico avvenuta in una scuola secondaria di primo grado. La seconda parte della tesi contiene il capitolo 4, dedicato alla progettazione e allo sviluppo dell'infrastruttura software di supporto. In tale capitolo saranno descritte l'architettura dell'infrastruttura e l'integrazione degli strumenti software.

Parte I

La ricorsione come tecnica per il problem solving

Capitolo 1

Ricorsione: stato dell'arte

Questo capitolo introduce i concetti base che caratterizzano il tema della ricorsione, per poi fare una rassegna di alcuni esempi didattici esistenti legati a questo tema. A tali esempi ci si è ispirati per la progettazione del percorso didattico che sarà descritto dettagliatamente nel capitolo 3.

1.1 Concetti base

La ricorsione è una tecnica che consente di risolvere un problema qualora questo presenti una particolare struttura. Si richiamano i concetti fondamentali alla base di questa tecnica, seguendo [1] e [2].

Per *problema*, qui intendiamo[1] una *funzione* tra l'insieme costituito dai possibili dati in ingresso (che possiamo definire *dominio*) e l'insieme delle possibili risposte (che possiamo definire *codominio*); per ogni dato in ingresso si ha esattamente una risposta al problema. Un'istanza di un problema è una coppia composta dal problema e da un dato in ingresso assegnato. Un algoritmo è un metodo generale che risolve un problema, ovvero riesce a determinare la risposta per ogni istanza del problema, attraverso una sequenza di passi computazionali. Un certo problema può essere risolto da algoritmi diversi.

Facciamo alcuni esempi. Calcolare la radice quadrata di un numero intero arbitrario è un problema. Un'istanza di questo problema è calcolare la radice quadrata di *9*. La risposta a questa istanza del problema è il numero

[1] Bisogna tenere conto che l'accezione di "problema" a cui si fa riferimento in questa tesi non è quella che comunemente viene usata nelle scuole primarie e secondarie. Infatti, in queste ultime si utilizzano generalmente le denominazioni "problema" e "classe di problemi" per riferirsi a quello che in questa tesi viene rispettivamente indicato con i termini "istanza" e "problema".

3. Prendiamo invece in considerazione il problema di invertire l'ordine dei caratteri di una stringa arbitraria. Un'istanza di questo problema è invertire l'ordine dei caratteri della stringa "mi piacciono i dolci". La risposta in questo caso è la stringa "iclod i onoiccaip im". Una soluzione a questo problema può essere rappresentata dal seguente algoritmo. Prendiamo la stringa in ingresso e usiamo due indici per indicare, rispettivamente, il primo carattere della stringa e l'ultimo (nel caso dell'istanza citata, 'm' e 'i'). Scambiamo i caratteri indicati dai due indici e facciamo avanzare, di una posizione, il primo indice verso destra e il secondo indice verso sinistra. Ripetiamo queste operazioni fino a quando i due indici indicheranno i due caratteri centrali della stringa (se il numero totale di caratteri fosse dispari, gli indici indicherebbero lo stesso carattere). Scambiando per l'ultima volta i caratteri indicati dagli indici, si otterrà la risposta all'istanza del problema.

La ricorsione è una tecnica che si può applicare quando gli elementi del dominio del problema hanno una struttura *autosimilare*, possono essere cioè descritti usando altri elementi, più piccoli, dello stesso dominio. Ad esempio, una stringa possiede una struttura autosimilare: essa può essere vista come l'unione tra un carattere e un'altra stringa, oppure come la concatenazione di due stringhe la cui lunghezza è pari alla metà di quella della stringa originale.

La ricorsione risolve un problema attraverso un'operazione di *riduzione*. Ridurre un problema X a un altro problema Y significa scrivere un algoritmo per X che usa un algoritmo per Y come se fosse una "scatola nera". La correttezza dell'algoritmo risultante non può dipendere da *come* funziona l'algoritmo per Y. L'unica cosa che possiamo assumere è che la "scatola nera" risolva Y correttamente. Per risolvere il problema X si *delega* dunque parte del lavoro alla soluzione del problema Y.

La ricorsione sfrutta un particolare caso di *riduzione* in cui un problema X viene ridotto allo stesso problema X, si parla infatti di *autoreferenzialità*. Più precisamente, per ogni istanza del problema:

- se questa è sufficientemente semplice, la si risolve (*caso base*);
- altrimenti, la si riduce a un'istanza più semplice.

Se l'autoreferenzialità può generare confusione, si può pensare che la soluzione delle istanze più semplici viene *delegata* ad altri soggetti.

Un algoritmo ricorsivo segue un approccio chiamato *divide et impera*: esso scompone il problema in *sotto-problemi* simili all'originale, per poi risolvere i sotto-problemi ricorsivamente. Formalmente, per sotto-problema P' di un problema P, intendiamo una *restrizione* della funzione definita da P ad un sotto-insieme degli elementi del dominio di P.

Il paradigma *divide et impera* prevede tre passi a ogni livello della ricorsione:

- dividi il problema in sotto-problemi;
- risolvi i sotto-problemi ricorsivamente;
- combina le soluzioni ai sotto-problemi nella soluzione al problema originale.

Prendiamo di nuovo in considerazione il problema di dover invertire l'ordine dei caratteri di una stringa arbitraria. Un algoritmo ricorsivo che risolve tale problema potrebbe essere il seguente. Si separa il primo carattere dal resto della stringa, si inverte il resto e infine si concatena il carattere alla porzione di stringa invertita. Nel momento in cui il resto della stringa sarà costituito da un solo carattere (il caso base), questo può essere considerato come già invertito, quindi non sarà necessaria nessuna operazione su di esso. Questo algoritmo risolve il problema adottando un approccio totalmente diverso dalla soluzione descritta precedentemente, poiché è definito in termini di sé stesso.

1.2 La didattica della ricorsione

La ricorsione è una tecnica che può essere usata nell'ambito del *problem solving*, espressione che in ambito informatico si riferisce in particolare alla capacità di trovare soluzione ai problemi mediante la progettazione e l'applicazione di algoritmi. Il *problem solving* è un processo creativo, che richiede a sua volta capacità di astrazione e pensiero computazionale e può dunque avere un importante valore formativo fin dalla scuola primaria e secondaria di primo grado.

Il *problem solving* è un processo mentale complesso [3] e anche se in alcuni casi un soggetto può sviluppare da solo delle strategie per affrontarlo in modo efficace (una strategia comune è ad esempio quella del *trial-and-error*), può essere di grande utilità seguire alcune strategie note al fine di ottenere più facilmente dei risultati. Generalmente, per risolvere un problema si inizia con la definizione dei requisiti e si termina con la definizione della soluzione, la quale in molti casi viene espressa con una sequenza di passi, ovvero con un algoritmo. Nel caso dell'informatica, spesso l'algoritmo viene espresso attraverso la codifica in un linguaggio di programmazione, che poi verrà testata con l'esecuzione del codice. Tutto ciò che sta tra la definizione dei requisiti e la formulazione della soluzione è un processo di scoperta, che può essere suddiviso in alcune fasi comunemente riconosciute:

- analisi del problema: vengono analizzate le caratteristiche del problema, identificando i dati che si hanno a disposizione in partenza e quelli che si vogliono ottenere alla fine, distinguendo le informazioni che possono essere utili da quelle irrilevanti; vengono inoltre cercate possibili analogie con casi già affrontati in passato;

- progettazione dell'algoritmo: viene definita la struttura generale del problema, identificando le parti che lo compongono e le relazioni tra queste, e vengono definite le strutture dati necessarie per modellare queste informazioni; vengono valutati approcci alternativi per affrontare il problema, ad esempio partendo dalle sue macro-componenti per poi considerare gli aspetti più specifici (approccio top-down), oppure il contrario (approccio bottom-up); nel caso il problema abbia delle forti analogie con problemi ricorrenti, ci si può affidare all'utilizzo di pattern progettuali;

- sviluppo dell'algoritmo: viene codificata effettivamente la soluzione;

- controllo della correttezza: viene controllato che la soluzione implementata sia corretta; questo concetto è legato sia ad aspetti teorici che ad aspetti più tecnici;

- riflessione: si riflette su quanto è stato prodotto per risolvere il problema e lo si analizza, prendendo in considerazione le prestazioni e l'accuratezza della propria soluzione, rendendo in questo modo più solide le proprie conoscenze al fine di essere in grado di risolvere in modo più efficace eventuali problemi futuri.

Una delle modalità più adatte per sviluppare capacità di *problem solving* è la didattica laboratoriale, poiché consente un maggior coinvolgimento e permette agli studenti di esplorare strategie per affrontare problemi, esprimendole concretamente. Inoltre fa comprendere in modo più profondo la natura degli algoritmi analizzati. La didattica laboratoriale mette in risalto gli aspetti scientifici dell'informatica, poiché fa mettere in pratica i suoi metodi scientifici di esplorazione, che consistono in un percorso di sperimentazione che inizia con la formulazione di un'ipotesi, continua con la sperimentazione e la raccolta dei dati, per poi concludersi con l'analisi dei dati e la formulazione delle conclusioni.

Una tecnica particolarmente singolare con cui si può affrontare il *problem solving* è quella della ricorsione, in quanto il suo approccio differisce fortemente da quelli più naturali e intuitivi. Insegnando questa tecnica a studenti di giovane età, ci si aspetta che vengano ampliati i loro orizzonti cognitivi.

Approcciare oggetti caratterizzati da struttura autosimilare con la tecnica della ricorsione fin da giovani, può aiutare ad sviluppare le proprie capacità di *problem solving* per riuscire ad analizzare i problemi guardandoli da una "prospettiva diversa". Prendiamo in considerazione, ad esempio, il problema di "salire le scale" [4]. Un esempio di procedura in linguaggio naturale che risolve questo problema è la seguente: fino a quando non sei in cima alla rampa di scale, sali un gradino. Adesso supponiamo di voler risolvere questo problema con una procedura ricorsiva. Un esempio di soluzione potrebbe essere la seguente. Se sei in cima alla rampa di scale, fermati; altrimenti, sali un gradino e poi *sali le scale*. Questa procedura risolve il suddetto problema richiamando sé stessa. Ciò è possibile grazie al fatto che fatto che una rampa di scale possiede una struttura autosimilare: una rampa di scale può essere vista come la successione di un gradino e una rampa di scale più corta.

La ricorsione rappresenta un tema affascinante da esplorare e questo può essere vantaggioso perché può fornire agli studenti una maggiore motivazione per affrontarlo. Un contesto particolarmente affascinante che si presta alla didattica è quello dei frattali. Un frattale è un oggetto geometrico che possiede una struttura autosimilare. La sua forma si ripete al suo interno su scale diverse, perciò se si effettua un ingrandimento di un'area qualsiasi di tale oggetto, si può vedere una forma simile a quella originale. Particolarmente interessante è la possibilità di disegnare frattali attraverso procedure ricorsive. Un primo esempio di questo approccio [5] utilizza LOGO, un linguaggio di programmazione per la grafica creato a scopo didattico da Seymour Papert (autore del termine "*costruzionismo*") in collaborazione con il MIT nel 1967; questo linguaggio nasce come dialetto del Lisp e presenta caratteristiche quali modularità, estensibilità, interattività e flessibilità. Un'altra proposta [6] utilizza Mathematica, un ambiente di calcolo simbolico e numerico. Entrambi sfruttano un interprete che riceve il codice sorgente e mostra graficamente il risultato prodotto; utilizzando linguaggi puramente testuali, però, presentano la difficoltà di dover affrontare una specifica sintassi senza il supporto di nessun altro elemento grafico. Sarebbe interessante invece riuscire ad affrontare questo lavoro con degli ambienti di programmazione visuali. Blockly [7] e Scratch [8] sono due buoni esempi, ma purtroppo non sono pensati per definire procedure ricorsive e pertanto non si prestano a questo scopo. Esiste anche un altro ambiente che consente di disegnare frattali: Recursive Drawing [9]. Questo ambiente non prevede l'implementazione di una procedura, ma consente di disegnare il frattale attraverso un'interfaccia grafica che richiede esclusivamente l'uso del mouse. Un esempio di questa interfaccia si può vedere nella Figura 1.1.
Questo ambiente permette di comprendere la struttura autosimilare della figura disegnata, anche senza richiedere alcun lavoro di programmazione.

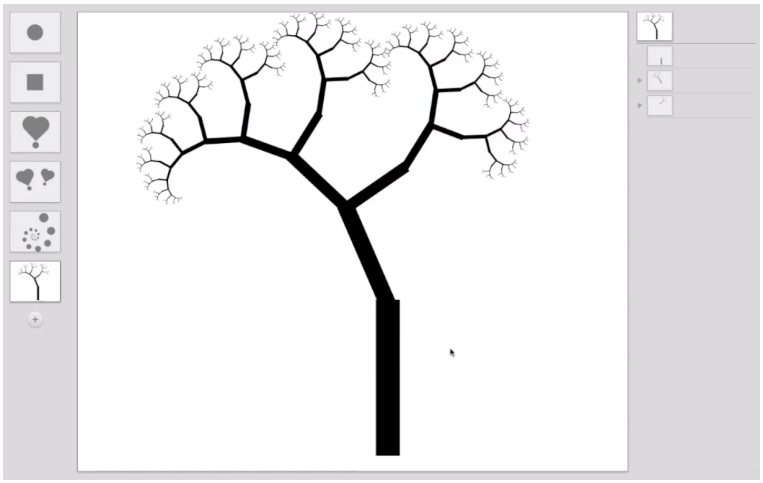


Figura 1.1: Screenshot della pagina web di Recursive Drawing

È bene infatti sottolineare che il *problem solving* non è necessariamente correlato alle capacità di implementazione (il *coding*, inteso come la traduzione di un algoritmo in un linguaggio di programmazione specifico). Infatti, per quanto sia un'attività complessa e creativa, il *coding* richiede di affrontare delle difficoltà legate prevalentemente alla sintassi del linguaggio utilizzato. Nel caso della ricorsione questo aspetto risulta secondario; la ricorsione può essere invece proposta con l'obiettivo di sviluppare pensiero computazionale, anche senza che sia previsto un lavoro di implementazione. Svolgendo invece un'attività di analisi dei problemi e di progettazione di un algoritmo, si mette in risalto la natura scientifica dell'informatica, poiché tale lavoro richiede un processo di scoperta che avviene attraverso la formulazione di ipotesi che devono essere validate da esperimenti [10].

In [11] si sostiene che, affrontando la ricorsione focalizzandosi sull'aspetto dell'implementazione (intesa come scrittura di codice sorgente o come compilazione in linguaggio macchina), si perdono di vista gli aspetti principali che caratterizzano questa tecnica. Infatti, diversamente dal modo in cui viene comunemente presentata a fini didattici, la ricorsione in realtà non ha niente a che vedere con uno "stack"; non prevede nessuna stretta dipendenza da funzioni, procedure o metodi. La ricorsione si basa principalmente sul concetto di autoreferenzialità: un algoritmo ricorsivo utilizza sé stesso per risolvere un problema ricorsivamente.

Si può prevedere, però, che l'autoreferenzialità possa generare confusione. La ricorsione, infatti, è caratterizzata dalla presenza di un meccanismo che inizialmente potrebbe risultare misterioso e di difficile comprensione; il suo apprendimento richiede un processo di intuizione che non è semplice da inne-

scare in modo deterministico. Inizialmente potrebbe essere necessario quello che talvolta viene definito "salto di fede" [3]: il primo passo per approcciarsi alla ricorsione è capire il nesso tra l'algoritmo e il suo output risultante, non tra l'algoritmo e il processo che mette in atto; quindi, bisogna prima capire come la procedura può essere descritta ricorsivamente e solo in seguito si potrà analizzare lo svolgimento del processo ricorsivo.

Si può inoltre osservare che la ricorsione racchiude un valore formativo anche quando si analizzano algoritmi senza svolgere un lavoro di progettazione. In tal caso, l'obiettivo diventa quello di comprendere il meccanismo che caratterizza tali algoritmi e diventare consapevoli che l'approccio funziona. Un esempio didattico esistente che cerca di raggiungere tale obiettivo, è rappresentato dai video realizzati da alcune persone della Sapientia Hungarian University of Transylvania. In questi video (che possono essere visti dal canale YouTube *AlgoRythmics* [12]), alcuni ballerini inscenano delle danze ungheresi in cui viene simulata l'esecuzione dei più comuni algoritmi di ordinamento, mostrando ogni passo del processo che questi algoritmi mettono in atto. Si potrebbe obiettare, però, che una semplice visualizzazione delle mosse prodotte da un algoritmo ricorsivo non sia il modo più efficace per comprenderlo. In questi video, infatti, si perde completamente di vista la struttura ricorsiva degli algoritmi eseguiti. Un'attività didattica che affronta la ricorsione, dovrebbe porsi come obiettivo quello di rendere essere più evidente e comprensibile tale struttura. In [3], ad esempio, sono descritte due attività che consentono di analizzare e comprendere l'esecuzione di una funzione ricorsiva: "The top-down frames model" e "The little people model".

La prima attività si basa sul tracciamento dell'esecuzione di una funzione ricorsiva attraverso il disegno di uno schema che descrive, per ogni livello della pila della ricorsione, l'ordine preciso delle istruzioni che vengono eseguite. In questo schema, per ogni chiamata della funzione ricorsiva viene disegnato un nuovo riquadro in cui vengono scritte le istruzioni eseguite in questa chiamata. Ogni riquadro è disegnato all'interno di quello della chiamata precedente. Terminato il disegno dello schema, tutti i riquadri saranno inclusi in un solo riquadro che corrisponde a quello della chiamata iniziale della funzione.

La seconda attività, invece, mostra l'esecuzione di una funzione ricorsiva attraverso la partecipazione attiva degli studenti, ai quali sarà richiesto di intervenire ad ogni nuova chiamata della funzione. La funzione a cui si fa riferimento nell'attività prevede l'esecuzione di istruzioni che stampano il valore del parametro in ingresso. L'attività si basa su una metafora secondo la quale, all'interno di un computer, esiste una vastissima comunità di piccole persone dove ognuna di queste è specializzata nell'esecuzione di una specifica funzione e più persone possono avere la stessa competenza. Quando una

funzione viene richiamata, lo "specialista" che è esperto nella sua esecuzione, si presta ad eseguirla indossando un vestito con delle tasche; il numero delle tasche corrisponde al numero degli argomenti della funzione eseguita. Ogni tasca può contenere il valore di un solo parametro. Inoltre, su ogni tasca è attaccata, sul lato interno, un'etichetta che corrisponde al nome del parametro. Gli studenti partecipano in questa attività impersonando ognuno un diverso "specialista". All'inizio dell'attività, un conduttore chiama alla lavagna uno studente, mette un foglietto con sopra scritto un numero (ad esempio '*3*') e gli dà in mano un gessetto. Da quel momento, lo studente dovrà eseguire le istruzioni previste dalla chiamata di funzione. Quando tali istruzioni sono quelle di stampa del parametro, dovrà eseguirle scrivendo alla lavagna il suo valore. Quando invece dovrà eseguire una nuova chiamata di funzione, richiederà a un compagno di eseguirla e resterà in attesa. Questo compagno andrà alla lavagna, gli sarà assegnato a sua volta il gessetto e gli sarà messo nella tasca un foglietto con scritto il parametro della nuova chiamata di funzione. Il processo continua in questo modo, fino a quando uno studente si troverà nel caso base della funzione ricorsiva. In quel momento si inverte la "direzione" del processo: questo studente, dopo aver eseguito l'istruzione di stampa, restituirà il gessetto al compagno che gli aveva richiesto di eseguire la chiamata di funzione, per poi tornare al proprio banco. La stessa cosa avverrà per tutti gli altri studenti alla lavagna che erano in attesa dell'esecuzione delle chiamate successive, fino a quando si completa lo svolgimento dell'intero processo.

A quest'ultima attività ci si è ispirati particolarmente per definire la modalità su cui si basano alcune attività del nostro percorso didattico.

Capitolo 2

Progettazione del percorso didattico

In questo capitolo illustriamo le considerazioni preliminari alla base della progettazione del percorso didattico, che riguardano in particolare la definizione degli obiettivi formativi, la gradualità delle attività proposte, il contesto in cui ambientare le attività, gli aspetti algomotori e quale uso prevedere del computer all'interno delle varie attività.

2.1 Obiettivi formativi

I principali obiettivi di questo lavoro di tesi sono stati la progettazione e lo sviluppo di un percorso didattico che affronta la ricorsione come tecnica per il *problem solving*. Prima di definire gli obiettivi del percorso didattico, abbiamo individuato questi quattro snodi cognitivi che riteniamo necessari per l'apprendimento della ricorsione:

1. comprendere il processo che viene messo in atto durante l'esecuzione di un algoritmo ricorsivo, quindi acquisire la consapevolezza di cosa avviene nei singoli passi di questo processo, ma anche avere una visione di insieme di come questi passi sono collegati tra loro;

2. dato un problema le cui istanze hanno una struttura autosimilare evidente o esplicitata, saper progettare un algoritmo che lo risolva ricorsivamente;

3. trovandosi di fronte a un problema caratterizzato da una struttura ricorsiva, essere in grado di riconoscere autonomamente questa struttura (senza formulare una soluzione per tale problema);

4. dato un problema le cui istanze hanno una struttura autosimilare non evidente, né esplicitata, saper riconoscere tale struttura e progettare un algoritmo che lo risolve.

Tenendo conto del livello di istruzione degli studenti a cui si è voluto rivolgere tale percorso, che corrisponde a quello del terzo anno della scuola secondaria di primo grado (ex terza media inferiore), abbiamo valutato che i primi due snodi fossero alla portata degli studenti con un lavoro iniziale di poche ore e su questi abbiamo quindi deciso di sviluppare il percorso. Gli altri snodi sono più complessi e necessitano di un lavoro ulteriore, per il quale abbiamo solo ipotizzato delle attività di massima.

Pertanto, gli obiettivi formativi del nostro percorso didattico possono essere così riassunti:

- l'acquisizione di conoscenze quali:
  - le caratteristiche principali della tecnica della ricorsione (vedi paragrafo 1.1);
  - il concetto di problema, inteso come "classe di problemi" (vedi la nota 1 del paragrafo 1.1);
- l'acquisizione di abilità quali:
  - saper eseguire una semplice procedura ricorsiva descritta in linguaggio naturale (ad esempio per calcolare la lunghezza di una stringa, per invertire una stringa e per calcolare una potenza di 2);
  - saper individuare le caratteristiche principali di un semplice algoritmo ricorsivo descritto e/o eseguito;
- l'acquisizione di competenze quali:
  - capire che un problema può essere risolto attraverso la risoluzione di sotto-problemi collegati fra loro.

Queste riflessioni sono state la premessa per la scelta dei contesti da affrontare e per la progettazione delle singole fasi del percorso didattico. Per una descrizione delle fasi progettate, si veda il paragrafo 3.1. Dei quattro snodi cognitivi descritti, il primo viene affrontato nella prima fase del percorso. Il secondo snodo viene affrontato, anche se non totalmente, nella quarta fase del percorso. Questo snodo cognitivo infatti è caratterizzato da una difficoltà notevole rispetto all'età degli studenti a cui il percorso è rivolto, pertanto, la

progettazione dell'algoritmo previsto è fortemente supportata dal conduttore delle attività. Il terzo snodo è toccato implicitamente nella seconda fase e ripreso nella terza. La terza fase aiuta a metabolizzare ulteriormente il percorrimento del primo snodo e del terzo.

2.2 Gradualità del percorso

Una questione delicata nella progettazione didattica è valutare la difficoltà di un compito assegnato agli studenti. In generale ci sono due aspetti da considerare, tra quelli che possono incidere sullo sforzo necessario allo svolgimento di tale compito:

- la difficoltà che caratterizza il tema a cui si riferisce il compito, unita al grado di complessità degli oggetti su cui gli studenti devono ragionare (ad esempio, le stringhe sono strutture più semplici degli alberi, oppure, ordinare un insieme di elementi è più difficile che contarli solamente);
- il tipo di compito che viene assegnato relativamente al tema e all'oggetto che sono stati scelti (ad esempio, in generale i compiti di tipo esecutivo sono più semplici di quelli creativi o progettuali).

Consideriamo ad esempio questi due compiti:

- eseguire i passi dell'algoritmo Insertion Sort [1] per ordinare una sequenza di numeri;
- progettare un algoritmo ricorsivo che conta gli elementi di una sequenza.

Nonostante il secondo compito sia legato a un tema più semplice, svolgere tale compito comporterà uno sforzo maggiore rispetto a svolgere il primo.

Una tecnica difficile quale la ricorsione, che presenta delle forti novità rispetto alle conoscenze pregresse degli studenti di una scuola secondaria di primo grado, per poter essere approcciata attraverso un percorso didattico necessita che le diverse attività che lo compongono presentino una proporzionata calibrazione dei due aspetti citati e che le attività prevedano un percorrimento graduale delle tappe previste. Tra un compito assegnato e quello successivo, il salto tra gli snodi cognitivi da percorrere non deve essere troppo alto, altrimenti gli studenti non raggiungeranno pienamente gli obiettivi formativi previsti. Gli studenti infatti partono da un bagaglio di conoscenze che può essere esteso svolgendo compiti di difficoltà superiore di volta in volta, ma solo se la differenza di difficoltà non è eccessiva. Questa estensione delle

conoscenze rappresenta il potenziale conoscitivo teorizzato da Lev Semënovič Vygotskij, che prende il nome di *zona di sviluppo prossimale*, o ZSP [13].

Una delle prime ipotesi durante la progettazione delle fasi del percorso era quella di assegnare agli studenti già nella prima fase il compito di progettare un algoritmo semplice (ad esempio, un algoritmo che inverte i caratteri di una stringa arbitraria), ma per le riflessioni appena illustrate si è ritenuto che assegnare tale compito così presto non avrebbe consentito agli studenti di raggiungere gli obiettivi, poiché in quel momento sarebbero stati privi degli strumenti mentali necessari. Si è quindi deciso che nelle prime fasi del percorso bisognasse assegnare loro dei compiti focalizzati esclusivamente sull'analisi e la comprensione di algoritmi già formulati, per poi far loro affrontare la progettazione di un algoritmo ricorsivo solamente nell'ultima fase del percorso.

2.3 Scelta del contesto

La scelta degli specifici contesti da usare per ambientare le attività didattiche è determinante per il raggiungimento da parte degli studenti degli obiettivi formativi prefissati. Perciò, per ogni contesto che si è preso in considerazione, si è ragionato su quali fossero gli obiettivi che avrebbe consentito di raggiungere, tenendo presente anche la modalità in cui è impostata l'attività didattica e l'età degli studenti a cui ci si rivolge.

Inizialmente si è preso in considerazione il gioco delle Torri di Hanoi. Questo gioco matematico prevede la presenza di tre paletti e di un determinato numero di dischi di grandezza diversa che possono essere infilati in uno qualsiasi di questi paletti. Quando il gioco inizia, tutti i dischi sono incolonnati sul primo dei tre paletti, in ordine decrescente di grandezza. Lo scopo del gioco è di portare tutti i dischi sul terzo paletto, potendo spostare solo un disco alla volta e potendo posizionare un disco solamente su un altro disco più grande. Il gioco è risolvibile con il seguente algoritmo ricorsivo [2]:

```
Hanoi(n, src, dst, tmp):
  IF n > 0 THEN:
    Hanoi(n - 1, src, tmp, dst)
    sposta il disco n da src a dst
    Hanoi(n - 1, tmp, dst, src)
```

Questo algoritmo è caratterizzato dalla presenza di una ricorsione multipla e poiché è in grado di risolvere questo gioco in modo molto efficace nonostante presenti delle operazioni semplici, potrebbe apparire particolarmente interessante al fine di mostrare il valore e le qualità dell'approccio ricorsi-

vo. Dopo alcune riflessioni riguardo alla complessità del processo che mette in atto l'esecuzione di questo algoritmo, però, si è pensato di prendere in considerazione dei contesti meno articolati che:

- fossero più vicini al grado di comprensione dello studente;
- facessero luce più facilmente sul processo che viene messo in atto dall'approccio ricorsivo, fornendo una maggiore consapevolezza riguardo al modo in cui questo processo si svolge;
- consentissero di interiorizzare in modo più solido l'approccio ricorsivo.

Successivamente ci si è concentrati sul contesto dei frattali. Come idea iniziale si è riflettuto su una possibile attività al pc che richiedesse di disegnare dei frattali attraverso la scrittura di procedure ricorsive in un linguaggio di programmazione per la grafica. Poi però si è preferito evitare di assegnare agli studenti un lavoro di implementazione, quindi si è ipotizzato un'attività che affrontasse i frattali in una modalità di tipo algomotorio, in cui fosse richiesta la realizzazione di alcuni frattali attraverso la composizione manuale di ritagli di carta. Ci si è accorti, però, che questa attività si sarebbe rivelata troppo dispersiva e che i compiti assegnati sarebbero stati troppo semplici. Quindi si è deciso di abbandonare tale contesto.

Un altro contesto che si è pensato si prestasse a essere affrontato con la ricorsione, è il calcolo delle potenze. L'idea era di affrontare questo contesto mettendo in risalto l'aspetto della delega, presentandolo agli studenti come "un modo per calcolare le potenze con meno fatica". Però si è ipotizzato che gli studenti potessero essere poco attratti da un contesto come questo, in quanto è di tipo strettamente matematico. Quindi l'idea è stata messa temporaneamente da parte.

In seguito ci si è concentrati su due contesti che potessero essere vicini alle conoscenze pregresse degli studenti:

- la ricerca binaria di una parola del dizionario;
- l'attraversamento di un albero n-ario.

Il primo contesto ci è sembrato potesse essere familiare agli studenti, poiché cercare una parola nel dizionario senza scorrerlo in modo lineare ci è sembrata un'operazione naturale per gli studenti dell'età presa in considerazione. Il secondo contesto si era pensato di affrontarlo prendendo in considerazione il caso del file system, precisamente considerando l'operazione di ricerca di un file all'interno di una cartella, o quella di leggere tutti i nomi di file e cartelle contenuti all'interno di tale cartella. Ciò che si è riscontrato analizzando

questi due contesti, però, è stato di non riuscire a immaginare delle attività efficaci e che bilanciassero il grado di semplicità del compito con la messa in evidenza della struttura autosimilare.

Infine ci si è concentrati su alcune operazioni che possono essere eseguite su una specifica struttura dati: la lista. In particolare:

- il calcolo della lunghezza;
- l'inversione;
- l'ordinamento degli elementi.

Si è osservato come questo contesto potesse riguardare anche la struttura dati della stringa, poiché essa può essere vista come una lista di caratteri. Questo contesto consente di:

- considerare il tema dell'elaborazione dei dati;
- considerare una struttura caratterizzata da diverse forme di autosimilarità (una stringa può essere vista come l'unione tra un carattere e un'altra stringa, oppure come la concatenazione di due stringhe la cui lunghezza è pari alla metà di quella della stringa originale);
- lavorare su degli elementi che presentano diverse caratteristiche (le lettere possono essere maiuscole o minuscole, vocali o consonanti, accentate o non accentate, etc.);
- lavorare su degli elementi (le lettere) che fanno parte di un insieme caratterizzato da uno specifico criterio di ordinamento (l'alfabeto);
- affrontare una varietà di problemi diversi, caratterizzati da diversi gradi di complessità;
- tradurre le operazioni sulla stringa in attività algomotorie in cui vengono manipolati fisicamente dei mattoncini LEGO, sui ognuno dei quali viene attaccata una lettera.

Considerati questi vantaggi, abbiamo scelto questo contesto per ambientare le attività che costituiscono il percorso didattico.

Relativamente alla possibilità di affrontare un algoritmo ricorsivo per l'ordinamento, inizialmente si è ipotizzato di affrontare il Merge Sort [1] per il problema di dover ordinare alfabeticamente i caratteri di una stringa. In particolare sono state ipotizzate alcune attività in cui si prevedeva l'uso da parte degli studenti di uno strumento software con l'obiettivo di analizzare il

processo che mette in atto questo algoritmo, per poi assegnare loro il compito di progettare la fase di *merge* in linguaggio umano. Riflettendo su questa ipotesi, però, si è constatato che il salto cognitivo che avrebbero dovuto affrontare sarebbe stato elevato, poiché la differenza di difficoltà presente tra un algoritmo più semplice (ad esempio un algoritmo ricorsivo che stabilisce la lunghezza di una stringa) e il Merge Sort sarebbe stata eccessiva. Per questi motivi si è abbandonata questa ipotesi e ci si è concentrati solo sulle operazioni del calcolo della lunghezza e l'inversione di una stringa.

2.4 Aspetti algomotori e uso del computer

Il termine "*algomotorio*" proviene dal laboratorio ALaDDIn (Laboratorio di Didattica e Divulgazione dell'Informatica) [14], che si occupa della divulgazione e della didattica dell'informatica. Questo laboratorio ha concepito, a partire dal 2008, diverse attività didattiche con l'obiettivo di introdurre gli studenti delle scuole secondarie italiane ad alcuni concetti informatici [15]. Queste attività prevedono che i temi informatici vengano affrontati con una modalità di tipo motorio (da qui nasce il termine "*algomotorio*"), in cui avviene una fisicizzazione di concetti astratti attraverso la manipolazione di oggetti e l'interazione con l'ambiente circostante. La maggior parte dei percorsi didattici di ALaDDIn è strutturata in modo da iniziare con una fase algomotoria e terminare con una fase al pc che prevede il consolidamento, attraverso l'uso di applicazioni software, dei modelli mentali costruiti dagli studenti durante le attività precedenti.

Questo approccio si ispira al *costruttivismo* [16] e al *modello di apprendimento allosterico* [17]. Il primo rappresenta una teoria secondo cui il soggetto che apprende costruisce la sua conoscenza riorganizzando e ridefinendo le sue conoscenze pregresse; lo fa gradualmente, basandosi sulle proprie strutture mentali esistenti e sui feedback che egli riceve dall'ambiente in cui apprende. Quindi le strutture mentali si sviluppano per gradi, elaborando quelle precedenti, infatti per questo motivo esiste anche la possibilità che la conoscenza compia delle regressioni o che percorra dei vicoli ciechi. Al fine di facilitare questo processo di costruzione graduale della conoscenza, bisogna fornire allo studente un ambiente di apprendimento in cui egli possa essere attivo (questo meccanismo viene appunto chiamato *apprendimento attivo*). Secondo l'allosterismo, invece, la trasmissione diretta della conoscenza è raramente l'approccio migliore per l'apprendimento. L'apprendimento viene visto come un processo attivo che avviene in modo conflittuale e integrativo tra ciò che lo studente ha nella sua mente e cosa egli può trovare e comprendere dall'ambiente circostante attraverso i suoi schemi mentali.

Per la progettazione delle attività didattiche [18], il laboratorio ALaDDIn si concentra innanzitutto sui principali obiettivi didattici che desidera che gli studenti raggiungano, per poi identificare una sequenza di passi intermedi che consentono agli studenti di costruire gradualmente la loro conoscenza. Queste attività vengono progettate in modo da essere stimolanti per gli studenti, così che siano motivati ad apprendere gli aspetti chiave il più possibile autonomamente e con la minor presenza possibile di indizi, perciò necessitano di essere chiare e di avere un senso di per sé, ad esempio grazie alla presenza di un contesto di drammatizzazione. Inoltre tali attività sono caratterizzate dal giusto equilibrio tra libertà di esplorazione dell'ambiente e presenza di vincoli esterni, così da consentire agli studenti il confronto reciproco e la possibilità di commettere errori, requisiti necessari per un'attività di *problem solving*. Infatti l'errore può fornire allo studente l'opportunità di correggere la propria conoscenza e migliorare la propria comprensione del concetto. Un'ulteriore strategia che viene usata per incentivare l'interesse degli studenti è quella di ricreare un contesto ludico. Infatti il gioco possiede un forte potenziale didattico, poiché consente di apprendere nuovi concetti in un'atmosfera alternativa che stimola le interazioni sociali e motiva fortemente la partecipazione attiva degli individui, specialmente quando scatena un meccanismo di competizione. Per questo motivo abbiamo deciso di basare la prima fase del percorso didattico su un'attività algomotoria in cui i caratteri delle stringhe (su cui verrà eseguito un algoritmo ricorsivo) sono rappresentati tramite dei mattoncini LEGO. Questo rende l'attività più concreta e maggiormente coinvolgente. Per la strutturazione del percorso didattico sulla ricorsione in diverse fasi ci si è ispirati alla struttura che caratterizza la maggior parte dei percorsi didattici del laboratorio ALaDDIn.

Un aspetto su cui si è riflettuto in fase di progettazione è stato l'approccio da adottare per la fase al computer. Sono stati delineati due differenti approcci. Il primo prevede che il sistema consenta di immettere o scegliere delle istruzioni che poi andranno a costituire un algoritmo; questo algoritmo poi verrà interpretato ed eseguito da un'applicazione e in base agli eventuali input immessi fornirà uno specifico output o mostrerà uno specifico comportamento. In questo sistema gli esperimenti vengono effettuati testando il programma implementato per stabilire se risolve correttamente il problema affrontato. Un esempio di questo approccio si trova nel percorso didattico *Labirinti* [10] e più in generale nelle proposte di *Code.org* [19]. Il secondo approccio invece prevede che il sistema presenti un algoritmo già implementato, più o meno articolato, per il quale si potranno scegliere i parametri da fornire in input. In base ai parametri scelti, il sistema potrà mostrare due diverse informazioni all'esterno:

- l'output finale, ovvero il risultato delle operazioni svolte dall'algoritmo;
- lo svolgimento del processo relativo all'esecuzione dell'algoritmo, attraverso un livello di astrazione specifico.

Quindi anche in questo caso il sistema consente di effettuare degli esperimenti, ma al fine di comprendere il funzionamento del processo svolto. Un uso di questo tipo del computer viene fatto ad esempio nel percorso didattico *Monete ed eventi* [15], in cui si affronta il tema degli algoritmi greedy. Per tale percorso è stato sviluppato uno strumento software ad hoc che consente di effettuare esperimenti per confrontare diversi algoritmi e analizzarne la correttezza. I due approcci descritti presentano una differenza sostanziale: mentre nel primo caso è richiesto che gli studenti svolgano un lavoro di progettazione e implementazione, nel secondo caso l'implementazione è già presente, non viene fornita una descrizione precisa di ciò che l'algoritmo svolge e si richiede gli studenti di svolgere un lavoro di analisi e di comprensione.

Per la progettazione dello strumento software usato nel percorso didattico sulla ricorsione ci si è avvalsi del secondo approccio, poiché per svolgere un lavoro di progettazione di un algoritmo ricorsivo sarebbero richieste delle competenze di cui gli studenti non dispongono ancora, nelle prime fasi del percorso didattico. Infatti si è optato per inserire un'attività focalizzata sullo svolgimento di tale lavoro solamente nell'ultima fase del percorso (la quarta). Inoltre, per la fase al computer si è scelto tale approccio anche per concentrarci maggiormente sull'aspetto dell'analisi, anziché su quello del "coding".

Nelle varie fasi del percorso didattico sulla ricorsione, per simulare l'esecuzione di un algoritmo ricorsivo ci si è ispirati fortemente all'attività "*The little people model*" (si veda il paragrafo 1.2). Per quanto riguarda la drammatizzazione dello scenario da mostrare nel software, si è optato per cambiare la figura degli "specialisti" usata in quell'attività con la figura delle "fatine". In questo modo l'attività può coinvolgere più facilmente anche gli studenti di genere femminile, innanzitutto perché non sarebbe rimarcato lo stereotipo secondo cui i problemi informatici vengono risolti principalmente da figure di genere maschile, ma anche perché il concetto di "magia" richiama un contesto più fiabesco, anziché tecnico.

Capitolo 3

Descrizione del percorso didattico e dei materiali predisposti

In questo capitolo si presentano il percorso didattico progettato per introdurre il tema della ricorsione in una classe terza di una scuola secondaria di primo grado. Nel paragrafo 3.1 si descrivono brevemente le fasi che costituiscono il percorso e le attività previste; nel paragrafo 3.2 si descrive lo strumento *software* sviluppato a supporto del percorso, analizzando nel dettaglio le scelte progettuali fatte; nel paragrafo 3.3 si illustrano alcune considerazioni didattiche e progettuali relative alle varie fasi del percorso; infine nel paragrafo 3.4 si riassumono gli esiti della sperimentazione del percorso didattico avvenuta in una scuola secondaria di primo grado, riportando le criticità riscontrate e i miglioramenti ipotizzati per il percorso e i materiali usati.

Il percorso didattico è composto da quattro fasi, volte al raggiungimento graduale degli obiettivi formativi prefissati:

- la prima fase è basata sull'esecuzione di un algoritmo ricorsivo per il problema del conteggio delle lettere di una parola, che avviene con modalità algomotoria;
- la seconda fase è basata sull'esplorazione e l'analisi di un algoritmo ricorsivo che serve a invertire una stringa; l'esplorazione avviene attraverso l'uso di uno strumento software sviluppato appositamente, ed è guidata da una scheda di lavoro;
- nella terza fase mira si confrontano gli algoritmi ricorsivi visti;
- la quarta fase mira a consolidare i concetti appresi proponendo la progettazione guidata di un algoritmo ricorsivo per il calcolo delle potenze; dopo la progettazione, l'algoritmo viene eseguito nuovamente in modalità algomotoria.

Il percorso è pensato per essere diretto da un conduttore principale, accompagnato dalla presenza di 1-2 tutor che lo aiutino nella conduzione delle attività a gruppi e nella gestione di eventuali problematiche secondarie. La durata prevista per lo svolgimento delle attività è di 3 ore circa. Precisamente, per le varie fasi del percorso è prevista un'ora per ciascuna delle prime due fasi e un'ora complessivamente per le ultime due fasi.

3.1 Dettaglio delle attività proposte

Saranno ora descritte dettagliatamente le attività che costituiscono il percorso, evidenziandone le finalità didattiche.

Prima fase: calcolare la lunghezza di una stringa con il LEGO

In questa fase gli studenti hanno il compito di eseguire, con modalità algomotoria, un algoritmo basato su una funzione ricorsiva che stabilisce la lunghezza di una parola. L'obiettivo è comprendere il funzionamento e le caratteristiche dell'algoritmo. Per concretizzare il lavoro degli studenti, la parola è rappresentata da una torre di mattoncini LEGO su cui sono scritte le lettere.

Gli studenti vengono suddivisi in gruppi di 5-6 e ogni gruppo viene fatto sedere in una fila di banchi. A ogni studente si consegna un foglietto con una descrizione, scritta in linguaggio naturale, della funzione ricorsiva. Il contenuto del foglietto è il seguente.

```
Quando ti si chiede di stabilire quanto è lunga una parola,
allora esegui queste istruzioni:
    Se la parola è formata da una sola lettera, allora
    sussurra al tuo compagno di destra il numero 1.
    Altrimenti:
    - stacca una lettera dalla parola e tieni la lettera staccata
    - passa il resto della parola al tuo compagno di sinistra
      chiedendogli di stabilire quanto è lunga
    - aspetta che il tuo compagno di sinistra ti sussurri la
      risposta all'orecchio
    - aggiungi 1 e sussurra il risultato a chi te l'aveva chiesto
```

È necessario specificare che dovranno eseguire queste istruzioni alla lettera, e che non dovranno preoccuparsi se inizialmente non è chiaro cosa stia accadendo, poiché sarà più chiaro in seguito. Non viene anticipato qual è il risultato finale che si dovrà ottenere.

Allo studente più a sinistra di ogni gruppo si consegna una torre di LEGO, chiedendogli di stabilire quanto è lunga la parola. Quando i gruppi hanno terminato di eseguire una prima volta l'algoritmo, si fanno altre prove cambiando ogni volta le posizioni degli studenti all'interno dello stesso gruppo. Le parole da assegnare ai vari gruppi sono di lunghezza diversa e dopo un certo numero di esecuzioni si scambiano le parole tra i vari gruppi. Una volta che gli studenti hanno svolto un numero di esecuzioni che si ritiene adeguato, si interrompe questa parte e il conduttore effettua insieme a tutta la classe un momento di riflessione su quanto è avvenuto prima. L'obiettivo è di verificare che gli studenti abbiano compreso cosa è avvenuto durante il processo che hanno messo in atto, mettendo in risalto alcuni punti chiave della ricorsione come tecnica per contare le lettere di una parola.

Seconda fase: analisi di un algoritmo ricorsivo supportata da uno strumento software

Durante la seconda fase del percorso didattico, si chiede agli alunni di analizzare il funzionamento di un algoritmo ricorsivo mediante l'utilizzo di uno strumento software progettato appositamente. Lo scopo del lavoro è comprendere la finalità e il funzionamento dell'algoritmo, che non sono resi noti agli alunni all'inizio dell'attività. L'attività è svolta a coppie, a ciascuna delle quali viene messo a disposizione un computer. Lo strumento software consente di eseguire l'algoritmo e di fare degli esperimenti intervenendo durante la sua esecuzione; tali esperimenti consentono di osservare aspetti caratteristici, e via via più specifici, del processo svolto e dunque di rilevare gli elementi necessari a comprendere il funzionamento dell'algoritmo.

Benché l'algoritmo sia prestabilito, il compito assegnato agli alunni in questa fase del percorso didattico non è puramente esecutivo come nella prima fase, ma è richiesto un lavoro di analisi e di astrazione. Per non rendere troppo difficile la consegna, si è dunque scelto un algoritmo molto simile, nella struttura, a quello della prima fase del percorso didattico. Tale algoritmo è definito da una funzione ricorsiva, riportata in Figura 3.1, che, avendo come argomento una stringa, la restituisce invertita (ovvero con i caratteri nell'ordine inverso, da destra verso sinistra).

L'interfaccia prevede tre modalità (chiamate *livelli*, per richiamare il linguaggio dei videogiochi) di visualizzazione e interazione. Ciascuno di questi livelli consente di osservare l'esecuzione dell'algoritmo da punti di vista di profondità crescente:

- il primo livello mira alla comprensione di quale sia l'algoritmo affrontato, ovvero l'inversione della stringa, e mostra semplicemente la correlazione tra l'input inserito e l'output restituito dall'intero algoritmo;

- il secondo mostra lo svolgimento del processo da un punto di vista globale, mettendo in risalto i seguenti aspetti:
  - la delega, che avviene tra le varie chiamate di funzione del processo;
  - gli argomenti che vengono passati in queste chiamate e i valori che vengono restituiti dai *return* conseguenti.
- il terzo mostra le istruzioni che vengono eseguite nel corpo delle singole chiamate di funzione, mettendo in evidenza che sono uguali per tutte le chiamate (escluso il caso base).

L'attività è guidata da una scheda composta da tre sezioni (una per livello) che, attraverso alcune domande, accompagnano gli alunni alla scoperta del funzionamento dell'algoritmo e delle sue componenti e/o caratteristiche fondamentali. Le domande sono formulate in modo da favorire negli alunni un lavoro di astrazione, poiché risulta difficile rispondere correttamente se si procede solo per tentativi, o se si esegue il programma osservandolo solo passivamente.

Terza fase: confronto tra gli algoritmi visti

Nella terza fase si confrontano i due algoritmi visti nelle fasi precedenti, ripercorrendone gli aspetti chiave e mettendo in relazione quelli in comune. All'inizio vengono riconsegnate le schede di lavoro compilate nella seconda fase, e vengono ripercorse le domande discutendo con gli studenti delle risposte. In questo momento ci si assicura che tutti abbiano chiaro in mente il funzionamento del software della seconda fase. A questo punto, insieme agli studenti, vengono rilevati gli elementi caratteristici degli algoritmi affrontati nelle prime due fasi, cercando somiglianze e differenze.

Dopo aver fatto emergere tali elementi, vengono distribuite nuovamente agli studenti le istruzioni usate nella prima attività e si rileva che esistono istruzioni simili, anche se non visibili, nell'elaboratore della seconda attività. Infine si chiede di ragionare su quali possano essere tali istruzioni e di provare a formalizzarle, chiedendo quindi agli studenti di svolgere una rielaborazione di quanto hanno appreso finora nel percorso didattico.

Alla fine dell'attività si mostra con un proiettore una porzione di codice Javascript che va a costituire una funzione ricorsiva che inverte una stringa. Il codice è stato scritto usando dei nomi che possono essere comprensibili dagli studenti. Questa funzione deve essere illustrata brevemente, senza entrare nel dettaglio. La porzione di codice che viene loro mostrata è quella nella Figura 3.1.

```javascript
function capovolgi_questa_frase( frase_da_capovolgere )
{
    if( frase_da_capovolgere.length == 1 )
    {
        return frase_da_capovolgere; // rimanda indietro la frase
    }
    else
    {
        var prima_lettera = frase_da_capovolgere.charAt(0); // stacca la prima lettera

        var resto_della_frase = frase_da_capovolgere.slice(1); // prende il resto della frase

        var resto_capovolto = capovolgi_questa_frase( resto_della_frase ); //richiesta a fatina successiva

        var frase_capovolta= resto_capovolto + prima_lettera; // attacca la lettera in fondo

        return frase_capovolta; // rimanda indietro la frase
    }
}
```

Figura 3.1: Screenshot della porzione di codice Javascript

Quarta fase: progettazione guidata di un algoritmo ricorsivo per il calcolo delle potenze

Nella quarta fase si guidano gli studenti nella progettazione di un algoritmo ricorsivo per calcolare la potenza 2^n, con n pari al numero degli studenti in classe. L'algoritmo risolve un problema caratterizzato da un contesto diverso da quelli affrontati nelle prime due fasi del percorso didattico: non si tratta più di elaborare una stringa, ma di calcolare una potenza.

Dopo aver fatto con gli studenti un breve riepilogo delle proprietà delle potenze, si cerca di formulare collettivamente un algoritmo ricorsivo che avrà una forma simile alla seguente:

```
Potenza (esponente):
  IF esponente == 1 THEN:
    return 2
  ELSE:
    risultato = Potenza(esponente - 1)
    return risultato * 2
```

Dopodiché gli studenti dovranno eseguire tale algoritmo con modalità algomotoria, eseguendo ognuno una chiamata di funzione e osservando lo svolgimento del processo.

3.2 Descrizione del software

Lo strumento software, usato nella seconda fase del percorso, consiste di una pagina web sviluppata in Javascript, facendo uso della libreria Raphaël [20]. Il lavoro di sviluppo non è stato particolarmente oneroso; la parte più

significativa del lavoro ha riguardato invece la progettazione e la messa a punto dell'interfaccia. Numerose sono state le modifiche e i ripensamenti, volti a rendere lo strumento più efficace dal punto di vista didattico.

In questo paragrafo descriviamo dettagliatamente la pagina web e le interazione previste, quindi mettiamo in evidenza le motivazioni didattiche alla base delle scelte progettuali, nonché gli accorgimenti e le modifiche principali che abbiamo introdotto durante la fase di sviluppo.

Come già detto, la pagina web consente di analizzare il funzionamento e le caratteristiche di un algoritmo basato su una funzione ricorsiva che riceve come argomento una stringa e restituisce la stringa capovolta.

Nella pagina web è presente un "elaboratore" sul cui schermo vengono mostrati degli oggetti che si muovono, concretizzando il processo che avviene al suo interno. Il software consente di scegliere la stringa da fornire in input all'elaboratore e di avviare l'elaborazione. Dopo l'esecuzione, l'interfaccia mostrerà la stringa ottenuta in output. L'elaborazione può essere avviata scegliendo uno di tre diversi livelli, i quali consentono di ispezionare ognuno un aspetto diverso del processo. Come in un videogioco, i livelli vengono sbloccati gradualmente: per sbloccare il livello 2 bisogna prima aver visto almeno una volta il livello 1 e per sbloccare il livello 3 bisogna prima aver visto almeno una volta il livello 2. Durante l'esecuzione dei livelli 2 e 3, la pagina web presenta dei comandi:

- un pulsante dedicato specificamente all'esplorazione del contesto, chiamato *ESPLORA*;
- un pulsante che consente di terminare anticipatamente l'elaborazione in corso;
- un regolatore per la velocità di esecuzione dell'elaborazione.

Sulla scocca dell'elaboratore è presente una clessidra che serve per illustrare il flusso del tempo durante lo svolgimento del processo.

All'interno dello schermo dell'elaboratore, a seconda del livello di gioco, vengono mostrate alcune informazioni relative allo stato del processo in svolgimento. Tali informazioni sono rappresentate visivamente mediante alcuni oggetti che si muovono nello schermo:

- la pila della ricorsione è rappresentata da una sequenza di cerchiolini, dove ogni cerchiolino corrisponde a una chiamata della funzione ricorsiva e si hanno tanti cerchiolini quanti solo i caratteri della stringa di input;

- la porzione di stringa passata come argomento delle chiamate è rappresentata da una torre gialla che si pone sotto al cerchiolino corrispondente alla chiamata di funzione e l'altezza della torre è proporzionale alla lunghezza di tale porzione;

- nella prima fase del processo (quando avvengono le chiamate ricorsive) la torre si muove da sinistra verso destra accorciandosi, mentre nella seconda fase (quando i valori restituiti vengono usati per comporre la soluzione) la torre torna indietro verso sinistra riallungandosi;

- la distinzione tra le due fasi del processo è mostrata colorando la sabbia della clessidra in giallo nella prima fase e in arancione nella seconda; mentre ci si trova nel caso base della ricorsione (quando la porzione di stringa è di lunghezza 1) la sabbia è rossa;

- la lettera iniziale della stringa argomento di una chiamata è rappresentata da un quadratino giallo che la torre "lascia" sotto al cerchiolino man mano che si sposta verso destra e che "recupera" mentre torna indietro verso sinistra;

- l'esecuzione di ciascuna funzione è svolta da una "fatina", che agisce sulle lettere che compongono la stringa argomento della funzione.

I livelli consentono di esplorare ognuno un aspetto diverso del processo. Nel primo livello di gioco, l'elaboratore è una "scatola nera" che riceve semplicemente la stringa in input e la restituisce invertita. Negli altri livelli sono visibili altre informazioni relative allo stato del processo in svolgimento, ed è possibile fare esperimenti cliccando sul pulsante *ESPLORA* (che metterà in pausa il processo). Terminato un esperimento, è possibile far ripartire il processo e fare altri esperimenti. Per favorire il lavoro di astrazione degli alunni, non è però consentito effettuare esperimenti relativi a chiamate di funzioni una immediatamente successiva all'altra.

Nel livello 2, cliccando su *ESPLORA*, l'elaboratore mostrerà due informazioni relative allo stato del processo in quel momento:

- la porzione di stringa contenuta nella torre;

- il numero di lettere che sono state staccare dalla stringa in input, pari all'altezza della pila della ricorsione.

La Figura 3.2 e la Figura 3.3 mostrano ciò che viene visualizzato nel livello 2 prima e dopo la pressione del pulsante *ESPLORA*.

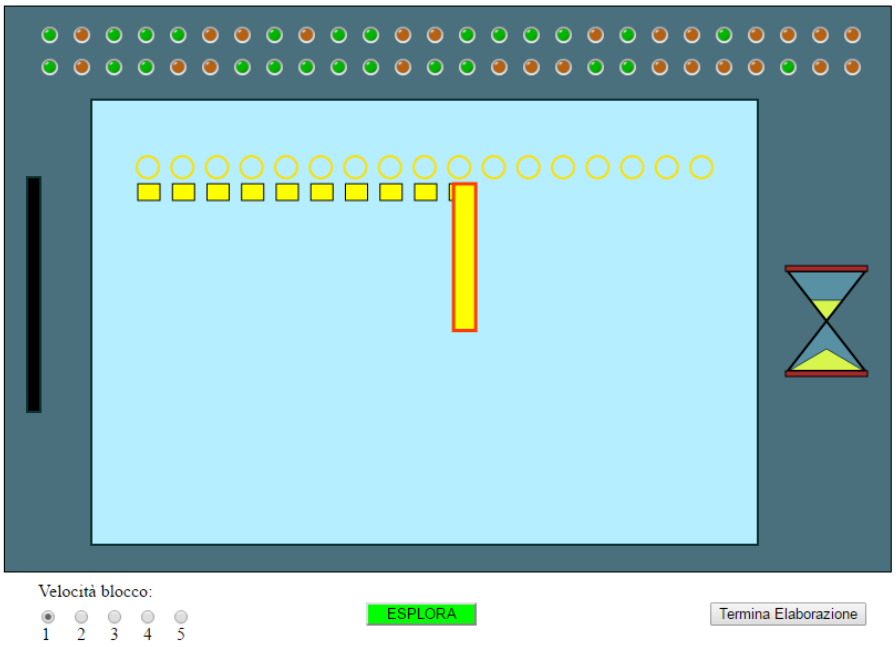


Figura 3.2: Screenshot del livello 2 prima della pressione di *ESPLORA*

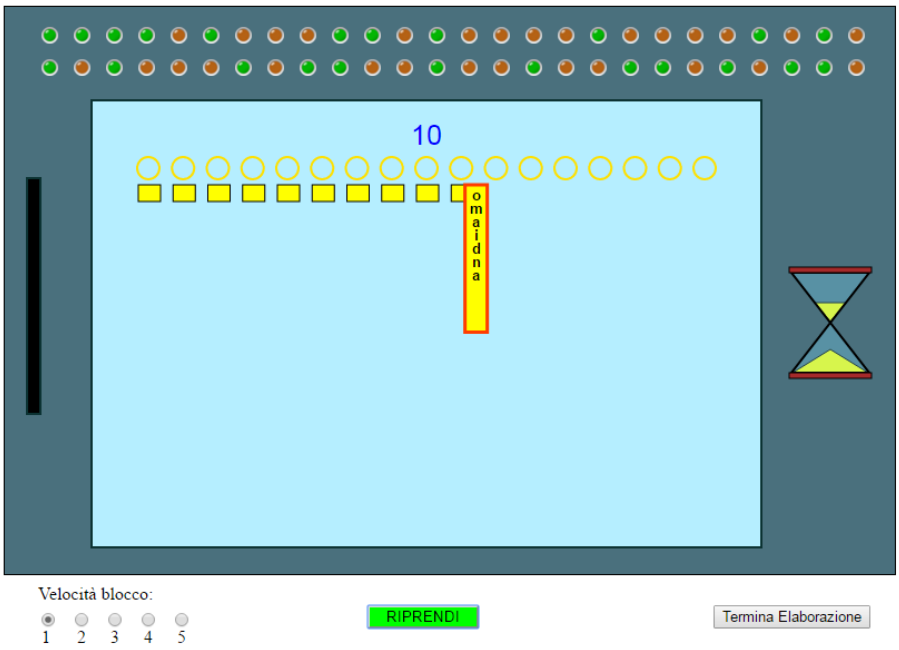


Figura 3.3: Screenshot del livello 2 dopo la pressione di *ESPLORA*

Cliccando su *ESPLORA* nel livello 3, invece, i cerchiolini diventano attivi; cliccando su un cerchiolino parte un'animazione che mostra in azione la fatina corrispondente a quel cerchiolino, assieme alle due fatine corrispondenti ai due cerchiolini adiacenti. Le azioni compiute dalle tre fatine dipendono dalla posizione delle fatine (o, più precisamente, dei cerchiolini ad esse corrispondenti) rispetto alla torre, nonché alla direzione della torre stessa. Si hanno i seguenti casi:

- se la torre è a sinistra della fatina, allora la fatina è addormentata;
- se la torre è a destra della fatina, allora la fatina ha una lettera in mano ed è in attesa;
- se la torre è esattamente sotto la fatina, allora la fatina agisce sulla torre e precisamente:
  - se la torre si sta muovendo verso destra, allora la fatina stacca una lettera dalla cima della torre e passa il resto della torre alla fatina che sta alla sua destra;
  - se la torre si sta muovendo verso sinistra, allora la fatina incolla alla base della torre la lettera che ha in mano, quindi passa la torre alla fatina che sta alla sua sinistra;
  - se la fatina è l'ultima a destra, allora la fatina passa la torre (che sarà composta da una sola lettera) alla fatina che sta alla sua sinistra.

Finita un'animazione è possibile cliccare su un altro cerchiolino, oppure far ripartire il processo. La Figura 3.4 mostra un esempio della schermata con le fatine in cui una di queste ha appena staccato una lettera dalla torre.

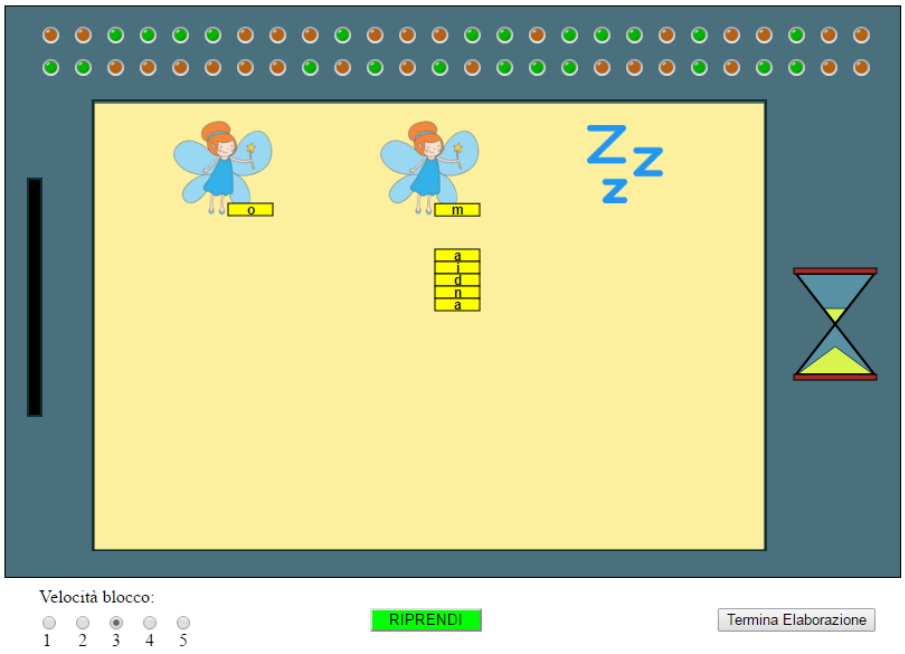


Figura 3.4: Screenshot della schermata con le fatine (le immagini delle fatine sono prese da Freepik)

3.2.1 Scelte progettuali

Lo strumento software descritto ha come scopo quello di facilitare l'analisi di un algoritmo ricorsivo già implementato e deve consentire di effettuare degli esperimenti al fine di comprendere il processo. In base a tali caratteristiche, si è riflettuto su quali interazioni bisognasse prevedere, con particolare attenzione a questi due aspetti:

- la scelta dei parametri da fornire in input all'algoritmo;
- la conduzione dei singoli esperimenti.

Più si dà la possibilità di personalizzare l'input dell'algoritmo e maggiori sono gli esperimenti che gli studenti possono effettuare. Di conseguenza è maggiore anche il grado di libertà che si fornisce agli studenti nell'esplorazione dell'algoritmo. Quindi si è cercato di raggiungere un compromesso tra la libera scelta dell'input e un'adatta visualizzazione del processo conseguente. Per riuscire a contenere tutti i cerchiolini nello schermo, si è deciso di imporre per l'input un numero massimo di caratteri consentiti. Per migliorare l'usabilità del software si possono scegliere degli input predefiniti.

Una conduzione più interattiva degli esperimenti può consentire un'esplorazione più efficace dei meccanismi interni all'algoritmo. Queste interazioni dipendono fortemente da quali informazioni mostra lo strumento software. In un primo momento si è pensato di rendere visibili soltanto alcune animazioni scelte casualmente riguardo al processo, ma ciò non avrebbe consentito una vera e propria esplorazione, perché la modalità di sperimentazione sarebbe stata troppo limitante. Quindi si è pensato di dare la possibilità di fare esperimenti più mirati. Consentendo una maggiore libertà di esplorare, maggiore sarebbe stato l'incentivo a progettare gli esperimenti invece di osservare passivamente il processo. Per rafforzare tale incentivo però, ci si è concentrati anche su quali vincoli fossero necessari relativamente alle interazioni per favorire un lavoro di astrazione da parte degli studenti e si è deciso di negare la possibilità di effettuare esperimenti relativi a chiamate di funzioni successive nel processo ricorsivo. Mentre inizialmente l'ispezione della torre e dei cerchiolini erano consentiti mentre la torre era in movimento, in seguito si è deciso di introdurre il pulsante *ESPLORA*, che mette in pausa il processo e consente di ispezionare quegli elementi quante volte si desidera in un determinato istante del processo. Infine, poiché un esperimento si sarebbe potuto rivelare lungo, si è voluto consentire la possibilità di interromperlo anticipatamente.

Per la scelta delle informazioni da mostrare nell'elaboratore durante l'esplorazione del processo, si è cercato di fornire suggerimenti essenziali e mai troppo espliciti, altrimenti sarebbe venuto meno l'approccio costruttivista relativamente alla comprensione dell'algoritmo. I suggerimenti forniti devono stimolare e guidare le esplorazioni. Inoltre, le informazioni mostrate non devono mai essere dispersive e devono mantenere coerenza con il processo. Anche il livello di astrazione con cui vengono illustrate tali informazioni è un aspetto su cui si è puntata l'attenzione, poiché in base all'età e alle conoscenze pregresse degli studenti determinerà il raggiungimento o meno degli obiettivi formativi prefissati. Quindi relativamente al processo ci si è concentrati sulla rappresentazione delle informazioni di tipo globale (relative alla pila della ricorsione e al "flusso" del processo) e di tipo locale (relative all'esecuzione delle singole funzioni): a questo scopo sono stati introdotti i livelli di gioco 2 e 3.

Negli esperimenti del livello 2 si sono voluti mettere in risalto gli input e gli output delle chiamate di funzione intermedie, infatti sulla torre è possibile vedere la porzione di stringa che viene passata come argomento alla chiamata di funzione corrente e in seguito i valori restituiti dai *return* che vengono usati per comporre il risultato finale. L'altra informazione che può essere vista in questo livello è il numero che indica l'altezza della pila della ricorsione, ovvero quante sono le fatine in attesa e quante sono le lettere che sono state già

staccate dalla stringa di input. Gli esperimenti del livello 3 invece consentono di porre l'attenzione su tutte le azioni che definiscono la funzione ricorsiva e che vengono compiute dalle singole fatine:

- separare la prima lettera dal resto della stringa ricevuta come argomento;
- passare la torre alla fatina adiacente a destra;
- attendere la torre restituita dalla fatina adiacente a destra;
- concatenare la lettera spezzata in coda alla porzione di stringa restituita dalla funzione successiva.

Nel livello 3 sono mostrate sia informazioni globali (relative all'intero processo), che locali (relative all'esecuzione di una singola funzione). In particolare, le azioni delle fatine dipendono dalla posizione e dalla direzione della torre rispetto al cerchiolino cliccato. Per rendere sempre disponibile il contesto, nel caso in cui la torre non si trovi sotto le tre fatine che compaiono nell'animazione, abbiamo aggiunto due elementi grafici:

- un piccolo segnalatore che sta a indicare se la torre in quel momento si trova più a destra o più a sinistra rispetto alle fatine visibili nell'animazione;
- una freccia che indica la direzione in cui è diretta la torre in quel momento.

Un altro aspetto su cui si è ragionato è stato il modo in cui avremmo dovuto visualizzare le lettere della stringa sulla torre. Queste avrebbero dovuto estendersi in modo adeguato nello spazio previsto dallo schermo dell'elaboratore. Inizialmente si è pensato di posizionare la torre in orizzontale: nel livello 2 ci sarebbe stata una schermata apposita per la lettura della porzione della stringa associata, mentre nel livello 3, nella schermata con le fatine avremmo mostrato in orizzontale solo le lettere della stringa limitrofe all'area su cui sta operando la fatina (il punto della separazione o della concatenazione della lettera). Con una rappresentazione grafica di questo tipo, però, non sarebbe più stato chiaro il momento del processo a cui ci si sta riferendo. Quindi si è deciso di posizionare la torre in verticale e di rendere visibili, nei livelli 2 e 3, tutte le lettere presenti sulla torre nel momento in cui si svolge l'esperimento, ovvero di mostrare esattamente la stringa argomento della funzione. Poiché sulla stringa da fornire in input all'elaboratore è presente un vincolo di lunghezza massima, si è riusciti comunque a dimensionare adeguatamente le lettere e ad adattare il layout mantenendo una buona leggibilità.

Per rendere più esplicito il fatto che le fatine tengono con sé la lettera che separano dalla porzione di stringa, nella schermata in cui la torre si sposta sotto i cerchiolini vengono visualizzati dei blocchetti gialli sotto ai cerchiolini relativi.

Per quanto riguarda gli aspetti di carattere prevalentemente scenografico, sono stati scelti degli oggetti significativi e graficamente accattivanti:

- la torre, come già detto, è l'oggetto che concretizza l'argomento delle chiamate di funzione ed è stato scelto per mantenere un punto di somiglianza con la prima fase del percorso didattico;
- le fatine, che vanno a eseguire le singole chiamate di funzione, sono state scelte per ricreare uno scenario "fiabesco" e per riuscire ad attrarre gli studenti di entrambi i generi;
- i cerchiolini sono stati scelti come elemento simbolico per non esplicitare fin da subito la presenza delle fatine e per stimolare l'esplorazione del processo nel livello 3; per attirare l'attenzione sui cerchiolini in fase di esplorazione, posizionandovi sopra il cursore del mouse, vengono illuminati i cerchiolini vicini a quell'area;
- la clessidra, la cui sabbia assume tre colori diversi, è stata scelta per illustrare il flusso del tempo durante il processo; mentre è gialla, la sua sabbia scende verso il basso, poi si capovolge dopo il caso base (durante il quale la sabbia è rossa), per poi scendere di nuovo mentre è arancione;
- dei led che si illuminano a intermittenza, sulla scocca dell'elaboratore, sono stati posizionati a scopo di enfatizzare l'elaborazione che sta avvenendo nell'elaboratore.

Questi aspetti, per quanto non siano strettamente collegati al contesto che si intende illustrare, racchiudono comunque un valore molto rilevante, perché servono a motivare gli studenti e incentivare il loro interesse (un fattore che va a incidere molto sull'efficacia di un'attività didattica).

Concludiamo questo paragrafo citando alcune altre ipotesi progettuali che poi sono state scartate:

- prima dell'introduzione della clessidra si era ipotizzato di usare la fila di led per illustrare il flusso temporale dello svolgimento del processo, ma si è optato per la clessidra poiché avrebbe espresso queste informazioni in modo più chiaro;
- prima dell'introduzione del pulsante *ESPLORA*, i suggerimenti del livello 2 sarebbero stati mostrati dopo un clic del mouse sulla torre e

sulla sequenza di cerchiolini, ma ciò rendeva difficoltosa l'interazione dell'utente;

- inizialmente si erano posizionati i cerchiolini in quantità prefissata e accavallati uno sull'altro, quindi non erano associati ciascuno a una singola chiamata di funzione; questo avrebbe reso poco chiaro cosa rappresentassero i cerchiolini;
- alla torre venivano fatti compiere dei piccoli balzi sotto alla sequenza di cerchiolini, ma tale animazione avrebbe solo generato confusione;
- inizialmente si era ipotizzato di associare i tre colori della clessidra anche alla torre, ma tali colori servono a rappresentare un'informazione globale del processo, mentre associati alla torre sarebbero potuti apparire come un attributo della torre stessa, quindi relativi alla funzione in esecuzione in quel momento;
- si era ipotizzato di mostrare le domande nella pagina web e di consentire la scrittura delle risposte al suo interno, ma ciò non era possibile per tutte le domande previste, quindi si è optato per la presenza di una scheda cartacea; in questo modo, però, le risposte non potranno essere corrette in modo automatico e non sarà possibile una loro archiviazione elettronica (utile per un'eventuale analisi dei risultati).

3.3 Considerazioni didattiche

Ripercorriamo ora le fasi di questo percorso per illustrare le considerazioni didattiche alla base delle scelte fatte in fase di progettazione del percorso stesso.

Prima fase

Per quanto il problema di contare le lettere di una parola sia un problema semplice, l'algoritmo analizzato si basa su un approccio che non rientra nelle conoscenze pregresse degli studenti. Non viene anticipato qual è il risultato finale che si dovrà ottenere, poiché altrimenti gli studenti sarebbero tentati di stabilire quanto è lunga la parola in una modalità più naturale e più vicina alla loro esperienza. Non si svela che il contenuto dei foglietti è identico per tutti gli studenti. È preferibile che siano convinti di avere ognuno un foglietto diverso, così da incentivarli a rivolgere la loro attenzione a tutto ciò che avviene nel processo. Comunque si può facilmente immaginare che già alla seconda prova si accorgeranno autonomamente dell'uguaglianza.

Cambiando ogni volta le posizioni degli studenti all'interno dello stesso gruppo e cambiando le parole tra i gruppi nelle diverse prove, essi avranno la possibilità di:

- osservare lo svolgimento del processo da prospettive diverse, cosa che sarà di particolare valore per gli studenti posti nella prima o nell'ultima posizione della fila (tali posizioni potrebbero essere identificate come quelle più cruciali);
- osservare più dettagli possibili, avendo a disposizione più tempo.
- concentrare la loro attenzione su ciò che svolgono singolarmente ma anche collettivamente;
- osservare il processo con istanze diverse del problema.

Chiaramente, nei casi in cui le parole hanno meno lettere rispetto al numero di componenti del gruppo, alcuni studenti non faranno nulla durante il processo.

Nella parte finale di questa fase si verbalizza a classe intera quanto è stato messo in atto dagli studenti. L'obiettivo è di verificare che gli studenti abbiano compreso cosa è avvenuto durante il processo, mettendo in risalto alcuni punti chiave della ricorsione come tecnica per contare le lettere di una parola. Invitando gli studenti a fare osservazioni e domande, si fa in modo che questi aspetti emergano da loro. Molto probabilmente emergerà che si erano accorti dell'uguaglianza delle liste di istruzioni. Il fatto che le istruzioni siano semplici e che uguali per ogni studente, può essere visto come un vantaggio; grazie alla collaborazione degli studenti, con queste istruzioni si può riuscire a eseguire un'operazione più complessa. Questa collaborazione si identifica con il concetto di delega: ogni studente risolve una parte del problema e delega il resto al compagno a fianco, fino a quando ci si trova nel caso base della funzione ricorsiva. Un altro aspetto che è utile far emergere, infatti, è l'indispensabilità del caso base. Lo studente che riceve una parola di lunghezza 1, è colui che inverte la "direzione" del processo: per quanto possa sembrare un'operazione semplice, in realtà è cruciale per lo svolgimento del processo.

Seconda fase

Il lavoro in questa fase è guidato da una scheda, suddivisa in tre sezioni, ognuna delle quali presenta delle domande relative a uno specifico livello dello strumento software. La prima sezione presenta anche un'introduzione che mira a chiarire il contesto, drammatizzarlo, così da incentivare l'interesse degli studenti e spiegare alcuni elementi dell'interfaccia grafica.

Le domande delle varie sezioni hanno lo scopo di:

- guidare gli studenti nell'accesso graduale ai diversi livelli;
- consentire l'osservazione di esempi concreti di funzionamento dell'algoritmo;
- indirizzare gli studenti nella progettazione degli esperimenti da effettuare;
- far riflettere gli studenti su ciò che hanno visto e fare in modo che emergano i loro dubbi, così da indurli a effettuare nuovi esperimenti;
- generalizzare le osservazioni fatte, arrivando ad una comprensione generale del processo in esecuzione.

Le nove domande accompagnano gli studenti alla scoperta del funzionamento dell'algoritmo e delle sue componenti fondamentali: la relazione tra l'input e l'output prodotto, il susseguirsi delle chiamate ricorsive e dei corrispondenti *return*, le stringhe passate come argomento alle funzioni, il caso base, le altre istruzioni del corpo della funzione ricorsiva. La scheda è riportata a pagina 76.

Nel dettaglio, gli obiettivi delle varie domande sono illustrati qui di seguito, facendo riferimento alla metafora delle fatine e della torre.

- La domanda 1 è l'unica che costituisce la prima sezione della scheda, legata al livello 1. Il suo obiettivo è di verificare che gli studenti abbiano capito qual è l'operazione svolta dall'elaboratore. La domanda è caratterizzata da un grado di difficoltà molto basso, ma assicura che gli studenti abbiano già effettuato degli esperimenti con lo strumento software prima di accedere alle domande successive.
- Con la domanda 2 ha inizio la seconda sezione della scheda (livello 2). Essa ha l'obiettivo di attirare l'attenzione degli studenti sulle modifiche che subisce la torre mentre scorre da un verso all'altro. In questo modo si cerca di far osservare che la porzione di stringa associata alla torre nella prima fase del processo si accorcia fino a ridursi ad una lettera, per poi riallungarsi nella seconda fase.
- La domanda 3 vuole attirare l'attenzione sul numero che compare durante le esplorazioni del livello 2. Ci si aspetta che gli studenti, dopo qualche esperimento, notino che il numero aumenta mentre la torre scorre verso destra e diminuisce mentre scorre verso sinistra.

- La domanda 4 serve a far osservare cosa succede quando ci si trova nel caso base, ovvero quando la clessidra è rossa. L'obiettivo è comprendere che in quel momento del processo la torre contiene soltanto l'ultimo carattere della stringa fornita in input all'algoritmo.

- La domanda 5 punta ad attirare l'attenzione sull'ordine in cui sono disposte le lettere quando la clessidra è arancione, ovvero in ordine inverso rispetto a come compaiono all'interno della stringa di input.

- Con la domanda 6 ha inizio la terza sezione della scheda (livello 3). Il suo obiettivo è di indurre gli studenti a osservare quali sono le operazioni che vengono eseguite dalle fatine sulla torre. Per la descrizione di queste operazioni, la domanda consente agli studenti un libero grado di dettaglio, anche perché la formulazione più precisa di queste primitive avviene nella terza fase del percorso didattico.

- La domanda 7 ha come obiettivo la comprensione di quali operazioni esegue una singola fatina quando riceve la torre, in particolare nella fase del processo in cui quest'ultima si muove verso destra (quando la clessidra è di colore giallo).

- La domanda 8 è simile alla precedente, ma si riferisce alla fase del processo in cui la torre si muove verso sinistra (quando la clessidra è di colore arancione).

- La domanda 9 è l'ultima della scheda e serve a indirizzare l'attenzione sulla correlazione esistente tra le fatine e le lettere della stringa in input. In particolare fa riferimento alla fase del processo in cui le fatine riattaccano le lettere alla torre (quindi dopo il caso base). Inoltre mira a comprendere che la torre subisce delle modifiche solamente dalla fatina posta vicina ad essa. Questa domanda richiede un processo di astrazione maggiore rispetto alle altre, poiché necessita che gli studenti abbiano compreso il processo globalmente e che riescano a immaginare l'elaborazione di una stringa arbitraria.

Alcune domande prendono in considerazione delle stringhe con un numero di caratteri inferiore al numero minimo consentito dallo strumento software. Questo accorgimento serve a evitare che gli studenti effettuino un esperimento con quelle stringhe precise (cosa che faciliterebbe eccessivamente il lavoro). Per riuscire a rispondere a tali domande è invece necessario che gli studenti progettino degli esperimenti adeguati, così da effettuare un'operazione di astrazione che consenta loro di generalizzare il processo.

I vocaboli usati nei testi della scheda di lavoro e della pagina web, sono stati scelti con l'obiettivo di consentire una lettura scorrevole da parte degli studenti, ma anche di fornire una descrizione precisa del contesto a cui si riferiscono.

Terza fase

Nel momento di confronto tra i due algoritmi proposti nelle prime due fasi ci si aspetta che emergano le seguenti somiglianze:

- gli algoritmi trasformano e fanno muovere una torre di mattoncini –*stringa*;
- una fila di agenti (gli studenti in un caso, le fatine nell'altro) collaborano tra loro; per ogni lettera della stringa si ha un agente –*ogni agente esegue una chiamata della funzione ricorsiva*;
- ogni agente (tranne l'ultimo) richiede ad un altro agente di risolvere il problema in questione su una porzione più piccola della torre –*delega, chiamata ricorsiva su un argomento più breve*– e attende la risposta per completare le sue istruzioni –*valore restituito, return*;
- gli agenti compiono tutti delle operazioni su una torre che può avere dimensioni diverse –*autosimilarità*;
- le istruzioni eseguite da studenti e fatine sono relativamente semplici, ma è la collaborazione di questi agenti che consente di risolvere un problema più complesso –*divide et impera*;
- disponendo di un numero arbitrario di soggetti che collaborano, l'algoritmo si adatta facilmente a una qualsiasi stringa –*vale per ogni istanza del problema*;
- l'ultimo studente e l'ultima fatina della sequenza ricevono una torre formata da un solo blocco e non agiscono su di essa, ma invertono solamente la "direzione" del processo –*caso base della ricorsione*;
- gli agenti eseguono tutti le stesse istruzioni, presenti anche nel secondo caso, anche se non visibili poiché le fatine le conoscono già –*corpo della funzione*.

Invece come differenze dovrà emergere che le operazioni effettuate sulla torre, nella fase del processo successiva al caso base, non sono le stesse in entrambi gli algoritmi. Infatti, mentre nella fase precedente al caso base gli agenti

staccano le lettere dalla stringa, nella fase successiva è solo nel secondo algoritmo che vengono riattaccate. Nel primo algoritmo infatti, grazie ai risultati provenienti dai *return* delle funzioni, gli studenti calcolano la lunghezza della stringa (valore che corrisponde a un numero anziché a una stringa). Nel secondo algoritmo, invece, le fatine restituiscono la torre, componendo gradualmente l'output finale dell'algoritmo, che corrisponde a una stringa (non a un numero).

Un aspetto fondamentale della ricorsione che non emerge nelle prime due fasi è quello dell'auto-referenzialità (una funzione ricorsiva richiama se stessa fino a quando ci si ritrova nel caso base). Questo aspetto non emerge poiché in entrambe le attività ogni agente delega l'esecuzione della chiamata di funzione successiva a un agente diverso. Inoltre non viene dato esplicitamente un nome alla funzione, pertanto non si vede che essa "chiama sé stessa".

Questo aspetto può essere invece sottolineato quando si mostra il codice JavaScript con le istruzioni per le fatine; gli studenti hanno modo di vedere un caso concreto di funzione ricorsiva implementata, che corrisponde all'algoritmo che hanno formulato relativamente alle azioni che svolgevano le fatine dello strumento software.

Quarta fase

Questa fase mira a consolidare ulteriormente i concetti emersi finora. Questo consolidamento viene favorito affrontando un contesto diverso da quelli delle prime due fasi, ovvero il calcolo delle potenze. Facendo progettare l'algoritmo agli studenti autonomamente, il salto tra gli snodi cognitivi percorsi fino a quel momento e il nuovo snodo sarebbe stato troppo elevato, non consentendo agli studenti di raggiungere l'obiettivo. Infatti un'attività di progettazione di un algoritmo è ben più difficile rispetto a un'attività in cui si analizza un algoritmo già formulato, inoltre l'oggetto dell'attività affrontata –le potenze– è più astratto rispetto a quelli su cui si basano le prime due fasi del percorso. Quindi questa attività di progettazione è guidata dal conduttore. Anche in questa fase vengono messi in risalto l'aspetto della delega e la struttura autosimilare del problema. È interessante notare che i primi studenti chiamati in causa nel processo dovranno in realtà calcolare gli ultimi passi della moltiplicazione e quindi dovranno affrontare i calcoli più difficili, cosa che potrebbe colpire gli studenti ed evidenziare la singolarità dell'approccio ricorsivo.

Questa fase è stata introdotta all'interno del percorso didattico in prossimità della sua sperimentazione, poiché inizialmente si pensava di concludere il percorso con la fase di confronto (la terza), ma poi è stato ritenuto utile introdurre anche questa fase in modo da percorrere anche il secondo snodo cognitivo citato nel paragrafo 3.4.3: dato un problema le cui istanze hanno una

struttura autosimilare evidente o esplicitata, saper progettare un algoritmo che lo risolva ricorsivamente.

3.4 Sperimentazione e revisione del percorso

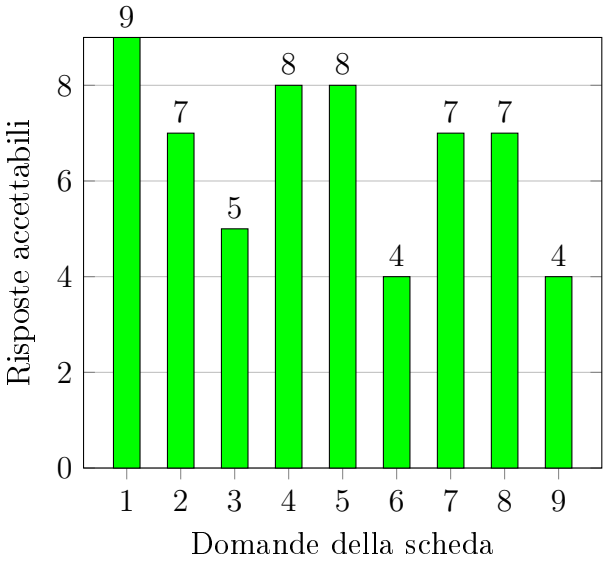
Terminata la fase di progettazione e sviluppo del percorso didattico e dei materiali, abbiamo svolto una prima sperimentazione sul campo, in una classe terza di una scuola secondaria di primo grado di Milano. La sperimentazione è avvenuta suddividendo le attività in due giorni consecutivi. Al termine di questa, è stato assegnato agli studenti della classe l'incarico di scrivere delle relazioni sulle attività a cui avevano partecipato. Segue un'analisi delle risposte sulle schede e delle relazioni scritte dagli studenti, al fine di fare una prima valutazione sull'andamento e sull'efficacia del percorso didattico sperimentato.

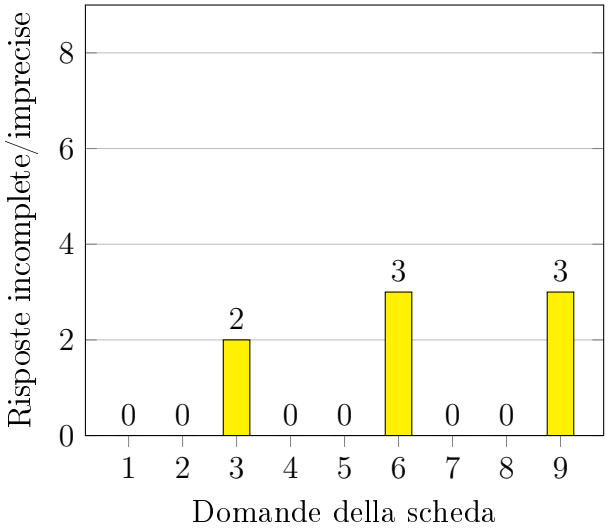
3.4.1 Analisi delle schede compilate

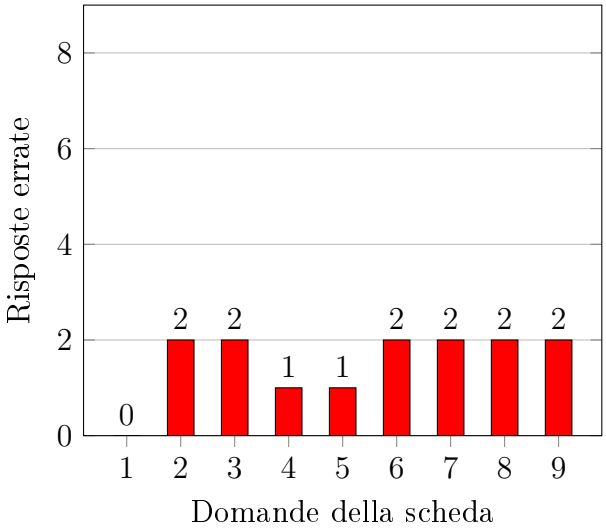
Come già è stato specificato nel paragrafo 3.1, la seconda attività viene svolta a coppie di studenti, a ognuna delle quali vengono assegnati un pc e una scheda di lavoro. A partecipare alla sperimentazione della suddetta attività erano presenti 18 studenti, che sono stati suddivisi quindi in 9 gruppi. La scheda è composta da 9 domande, suddivise in 3 sezioni diverse. Nel corso dell'attività, le diverse sezioni sono state consegnate agli studenti gradualmente, solo dopo aver verificato che la sezione consegnata precedentemente contenesse delle risposte che dimostravano un grado accettabile di comprensione degli aspetti trattati. Nei casi in cui le risposte erano ritenute troppo distanti da quelle auspicate, si è cercato di indirizzare gli studenti sulla strada più corretta fornendogli degli spunti di riflessione, ma senza suggerimenti espliciti.

I seguenti istogrammi illustrano, per ognuna delle domande della scheda, le risposte degli studenti, indicando:

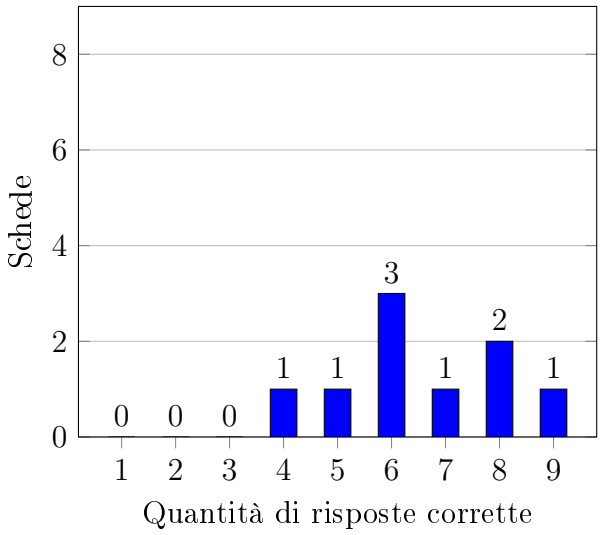
- in verde la quantità di quelle corrette o che si avvicinano a quelle auspicate;
- in giallo la quantità di quelle ritenute incomplete o imprecise;
- in rosso la quantità di quelle ritenute errate.


Risposte accettabili
0
2
4
6
8
9
7
5
8
8
4
7
7
4
1
2
3
4
5
6
7
8
9
Domande della scheda


Risposte incomplete/imprecise
0
2
4
6
8
0
0
2
0
0
3
0
0
3
1
2
3
4
5
6
7
8
9
Domande della scheda



Il seguente istogramma invece illustra la distribuzione degli studenti in base al numero delle risposte a cui hanno risposto correttamente.



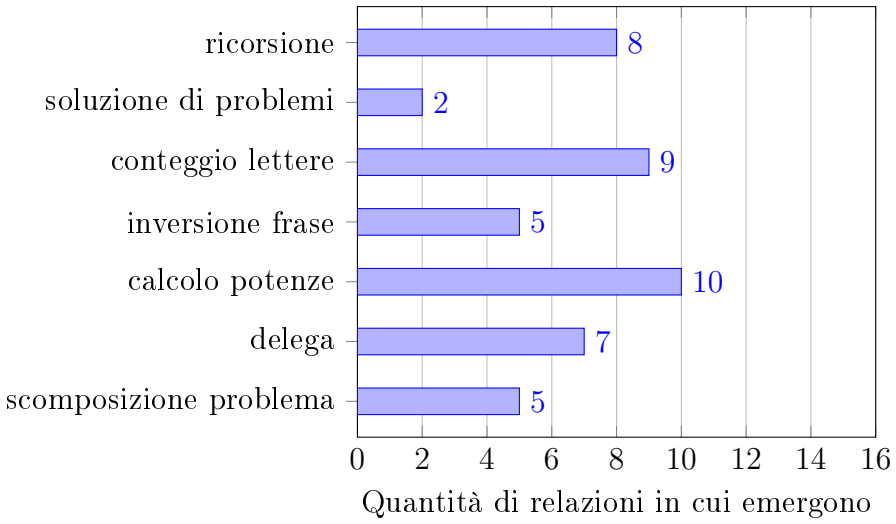
Da questi grafici si può osservare che le schede compilate dagli studenti nella seconda fase presentano un tasso di risposte corrette piuttosto elevato. In particolare, dall'ultimo grafico è possibile osservare come la distribuzione degli studenti si avvicini a una distribuzione di tipo gaussiano, con un valore medio sopra al 6. Le risposte alle domande 6 e 9 presentano un elevato tasso di errore; questo dato potrebbe dipendere dal fatto che queste domande siano caratterizzate da un alto grado di difficoltà, oppure da una scarsa chiarezza nella consegna.

3.4.2 Analisi delle relazioni degli studenti

Le relazioni scritte dagli studenti sono in totale 16, e contengono una loro descrizione del percorso didattico a cui hanno partecipato. Sono state analizzate prendendo in considerazione i concetti che emergevano da esse, tracciandone un resoconto statistico. Entrando nel merito delle relazioni, si è presa in considerazione la presenza dei seguenti concetti:

- il termine "ricorsione";
- il fatto che ci si è occupati di risolvere dei problemi;
- il fatto che sono stati affrontati 3 problemi diversi, ovvero il conteggio delle lettere di una parola, l'inversione di una frase, e il calcolo di una potenza. In particolare, si è preso in considerazione quali di questi 3 problemi erano specificati nelle relazioni;
- il concetto della delega, che gli studenti hanno espresso associandolo al concetto della collaborazione tra gli individui che lavoravano nei 3 processi affrontati;
- la scomposizione del problema in sotto-problemi più semplici.

Il seguente istogramma illustra, per ognuno dei concetti emersi, la quantità di relazioni in cui sono presenti tali concetti rispetto al totale.



Poiché la stesura di tali relazioni è stata assegnata senza specificare precisamente quali aspetti andassero trattati, lasciando così un certo grado di

libertà nella descrizione, non è possibile dedurre in modo diretto l'incomprensione di determinati aspetti sulla base di ciò che non emerge dalle schede. Infatti bisogna tenere presente che molte di queste relazioni presentano una descrizione molto generica delle attività svolte. Il termine "ricorsione" è comparso solamente in metà delle relazioni; questo non rappresenta necessariamente un problema, poiché durante l'attività non si è insistito sull'uso di questa parola, essendo invece l'obiettivo quello di presentare le caratteristiche principali di tale tecnica e lo svolgimento del processo che mette in atto, affrontando in particolar modo alcuni esempi di problemi.

Al termine della sperimentazione è stata fatta una breve intervista a Martina, l'insegnante della classe. Riportiamo qui il suo commento:

> In generale ho notato che i ragazzi sono stati contenti di aver partecipato alle attività previste dal percorso didattico. Nonostante la sperimentazione sia stata piuttosto estemporanea, le attività hanno sicuramente coinvolto gli studenti e hanno fornito loro un modo per vedere le cose da un punto di vista alternativo, attraverso una tecnica singolare e sicuramente nuova per loro. Sono contenta che abbiano potuto approfittare dell'occasione. La scrittura della relazione, seguente alla sperimentazione, è servita a farli riflettere su quanto intravisto, così da farlo sedimentare. Sarebbe stato opportuno approfondire l'argomento in classe nei giorni successivi, ma purtroppo non ce n'è stata la possibilità. Ho ritenuto efficace, come nelle altre attività del laboratorio ALaDDIn, l'abbinamento di una fase iniziale motoria a una che prevede l'uso di applicazioni software, per affrontare il tema del *problem solving*.

3.4.3 Criticità riscontrate e possibili miglioramenti

A seguito di un'analisi dell'andamento e dei risultati relativi alla sperimentazione descritta nel paragrafo 3.4, sono emerse delle ipotesi di modifiche da apportare al percorso didattico, quindi alle singole fasi che lo compongono e ai materiali usati.

Per quanto riguarda l'attività algomotoria della prima fase, si è notato che quando veniva richiesto agli studenti di stabilire quanto era lunga la parola, alcuni provassero l'impulso di dare una risposta immediata, anziché eseguire le istruzioni assegnate. Come soluzione a questo inconveniente sono stati ipotizzati i seguenti accorgimenti:

- agli studenti deve essere esplicitato in modo chiaro che devono attenersi rigidamente alle istruzioni assegnate, eseguendole alla lettera;

- le liste di istruzioni devono essere stampate con un'impaginazione migliore, così da renderne più scorrevole la lettura; inoltre devono essere stampate con 3 o 4 impaginazioni diverse, così da illudere gli studenti, almeno all'inizio, del fatto che il contenuto dei foglietti sia diverso.

Inoltre, per evitare che gli studenti restituiscano la torre di LEGO nella fase successiva al caso base, si è ipotizzato che sulla fila di banchi di ogni gruppo debba essere posizionato un piccolo cestino, nel quale ogni studente dovrà gettare la lettera dopo averla staccata dalla torre.

Per quanto riguarda lo strumento software usato nella seconda fase, invece, poiché è sembrato che non tutti gli elementi dell'interfaccia grafica venissero usati dagli studenti, si è ritenuto opportuno che a inizio attività debba essere chiarita in modo appropriato l'interfaccia della pagina web, in particolare citando la presenza dei tre livelli. Inoltre si è notato che talvolta i livelli venivano avviati frettolosamente, perciò si è ritenuto opportuno chiarire che i livelli 2 e 3 debbano essere avviati solo dopo aver ricevuto le relative sezioni della scheda. Invece, relativamente a ciò che mostra l'elaboratore presente nella pagina web, sono state ipotizzate le seguenti modifiche:

- per rendere inizialmente meno evidente l'operazione che avviene all'interno dell'elaboratore, le frasi predefinite da scegliere come input sarebbero dovute essere già capovolte, così da uscire in output nel verso corretto;

- per rendere più chiaro il passaggio della torre come argomento delle chiamate di funzione, le fatine avrebbero dovuto passare fisicamente la torre alla compagna, dopo aver staccato o riattaccato la lettera;

- per rendere più chiara la presenza dei tre livelli, si è ipotizzato di rendere i tre pulsanti sempre visibili ma di abilitarne la pressione solo nel momento in cui si sbloccano (inizialmente i pulsanti diventavano visibili solamente dopo lo sblocco dei livelli associati);

- per rendere più chiaro a cosa ci si riferisce quando si parla della "torre", si è ipotizzato di marcare il bordo di questa con un colore diverso rispetto ai blocchetti gialli che vengono lasciati sotto ai cerchiolini (in base alle risposte degli studenti sulla scheda di lavoro, si è notato che alcuni credevano che con il termine "torre" si intendesse l'insieme di tutti i blocchi gialli visibili sullo schermo, pensandoli come una cosa sola).

Infine sono state ipotizzate anche alcune modifiche per quanto riguarda la scheda di lavoro:

- per rendere più chiara la richiesta della domanda 2, si è pensato di sostituire "come cambia la torre" con "cosa succede alla torre";

- per rendere più chiara la richiesta della domanda 6, si è pensato di sostituire "che tipo di operazioni sono capaci di svolgere le fatine" con "che tipo di operazioni è capace di svolgere ciascuna fatina"; in questo modo è più esplicito il fatto che ci si riferisce al contesto locale della fatina e non alle fatine collettivamente;

- per rendere più pratica la scrittura del colore della clessidra nella domanda 9, si è pensato di inserire un apposito campo di testo.

Ragionando sul percorso didattico nella sua interezza, sono stati ipotizzati due possibili ampliamenti del percorso con due nuove fasi. Un possibile sviluppo richiede agli studenti di progettare, a gruppi di 2/3, un algoritmo per la ricerca di una lettera in una sequenza ordinata di lettere, partendo dallo spunto del dizionario. In questo modo si affronterebbe il quarto snodo cognitivo descritto nel paragrafo , quello relativo alla progettazione di un algoritmo per un problema di cui non viene messa in evidenza la struttura autosimilare. Si è scelto questo contesto poiché cercare una parola nel dizionario senza scorrerlo in modo lineare ci è sembrata un'operazione naturale per gli studenti dell'età presa in considerazione.

Un altro possibile sviluppo, rivolto però a studenti con una capacità di astrazione più elevata rispetto a quelli a cui ci si è rivolti finora (ad esempio se frequentassero una scuola secondaria di secondo grado), richiede di analizzare l'esecuzione dell'algoritmo *Merge Sort* per risolvere il problema di ordinare alfabeticamente i caratteri di una stringa, con uno strumento software analogo a quello con le fatine. In questo modo si approfondirebbe il primo snodo cognitivo del paragrafo , ma analizzando un noto algoritmo, più complesso e caratterizzato da un'altro tipo di struttura autosimilare. Questo algoritmo può anche fornire lo spunto per ragionare sull'efficienza computazionale di diversi algoritmi che risolvono lo stesso problema.

Parte II

Infrastruttura software di supporto

Capitolo 4

Infrastruttura di supporto

In seguito allo sviluppo del percorso didattico e a una sua prima sperimentazione, si sono individuati due nuovi obiettivi:

- riuscire a tracciare delle informazioni legate all'utilizzo da parte degli studenti del software associato, al fine di effettuare delle analisi statistiche e quindi di poter formulare un resoconto sull'efficacia delle attività dal punto di vista didattico, con la conseguente possibilità di apportare eventuali migliorie;
- rendere il percorso didattico fruibile anche ad altri docenti, semplificando la modalità di utilizzo e di installazione del software così da non dover richiedere particolari competenze.

Per questo motivo si è ipotizzato di progettare un'infrastruttura software di supporto che consentisse di raggiungere tali obiettivi. Riflettendo su questa ipotesi, si è constatato che un'infrastruttura di questo tipo si sarebbe prestata facilmente alla gestione di software didattici con caratteristiche simili, perciò si è pensato di valorizzare maggiormente questo lavoro destinando tale infrastruttura anche ad altri software esistenti e futuri del laboratorio ALaDDIn (si veda il paragrafo 2.4). Perciò agli obiettivi precedenti sono stati aggiunti quelli che seguono:

- rendere disponibile in una sola posizione locale l'intera suite di software didattici legati alle attività ALaDDIn;
- rendere l'infrastruttura fruibile indipendentemente dal sistema operativo presente sui computer usati, e indipendentemente dall'accessibilità della rete esterna;
- rendere rapida l'integrazione di eventuali nuovi software didattici, senza richiedere un numero eccessivo di adattamenti.

Per fare in modo che l'infrastruttura potesse supportare software didattici diversi, si sono presi in considerazione gli aspetti comuni dei software a supporto dei laboratori attualmente erogati, per poi astrarre le caratteristiche astratte di un modello di software didattico. Questo processo di analisi ha portato a individuare le seguenti peculiarità di un generico modello:

- il software didattico viene eseguito da gruppi di studenti;
- il software didattico presenta un'interfaccia utente con la quale degli studenti effettuano degli esperimenti legati a un determinato contesto e a un determinato problema;
- un esperimento si avvia con lo scatenarsi di uno specifico evento, che corrisponde all'interazione con un elemento dell'interfaccia;
- nell'interfaccia possono essere presenti degli elementi che consentono a chi compie gli esperimenti di specificare un input, oppure di impostare dei parametri che servono a configurare lo svolgimento dell'esperimento (questi parametri vengono rappresentati da elementi attivi dell'interfaccia del software);
- al termine dell'esperimento, il software produce tipicamente un risultato in output;
- il software potrebbe prevedere differenti livelli di utilizzo;
- nel caso in cui il software preveda che lo studente fornisca una soluzione a un problema, se si volesse verificare la correttezza di tale soluzione, è possibile che sia necessario disporre di alcune informazioni aggiuntive.

Partendo da questi presupposti, ci si è concentrati sul determinare quali specifiche informazioni risultasse interessante tracciare al fine di effettuare le analisi statistiche sopra menzionate, stabilendo che fossero particolarmente rilevanti le seguenti informazioni:

- alcuni dati anagrafici legati agli studenti che stanno utilizzando il software (l'età e il sesso di ogni studente);
- eventuali input legati all'esperimento e parametri che servono a configurare il suo svolgimento;
- un riferimento a quale esperimento viene tracciato;
- un riferimento temporale al momento in cui è stato svolto l'esperimento.

Va tenuto conto del fatto che al variare del tipo di esperimento gli input e i parametri di configurazione possono avere diverso valore statistico. Pertanto ogni specifico software da integrare nell'infrastruttura provvederà a tracciare solamente le informazioni ritenute significative.

Un aspetto separato, ma non indipendente dal resto, è invece rappresentato dai criteri usati per analizzare i dati raccolti, al fine di definire un resoconto sull'efficacia delle attività. Ad esempio, si potrebbe ottenere la frequenza con cui gli studenti hanno effettuato gli esperimenti in una specifica attività, per determinare il loro tasso di interazione con il software. In ogni caso, una volta scelto un appropriato insieme di dati rilevanti da tracciare, la definizione di tali criteri è un aspetto che può avere luogo in un secondo momento.

4.1 Tecnologie utilizzate

Per lo sviluppo dell'infrastruttura software si è optato per sfruttare la piattaforma Docker, in modo da rendere la fruizione dell'infrastruttura indipendente dalle tecnologie necessarie all'esecuzione dei software didattici. Inoltre in questo modo si ha il vantaggio di avere delle istanze software modulari che possono essere avviate al momento del bisogno, contenenti tutti i componenti necessari al funzionamento del software didattico associato. Per quanto riguarda l'archiviazione dei dati tracciati dall'infrastruttura, si è optato per sfruttare un DBMS NoSQL: MongoDB.

4.1.1 Docker

Docker [21] è una piattaforma open-source che consente di automatizzare l'installazione e l'esecuzione di un'applicazione in un ambiente che prende il nome di "container". Questa piattaforma sfrutta le funzionalità di isolamento delle risorse del kernel Linux (a partire dalla versione 2.6.24) e un file system union-capable per consentire a container indipendenti di essere eseguiti all'interno di un'unica istanza del sistema operativo, evitando un eccessivo sfruttamento di risorse che sarebbe invece richiesto dall'avvio e dalla gestione di una macchina virtuale. Il supporto dei namespace del kernel Linux riesce in buona parte a isolare le interazioni tra l'applicazione e il sistema operativo, inclusi componenti come l'albero dei processi, la rete, gli user ID e i file system montati; inoltre il kernel si occupa di limitare l'uso di risorse quali CPU, memoria, periferiche I/O e rete.

Dal punto di vista architetturale, Docker è un sistema di tipo client-server. Il client comunica con un demone che si occupa di costruire, eseguire

e distribuire i container. Sia il client che il demone possono essere eseguiti in un host che risiede sullo stesso sistema, oppure è possibile connettere il client con un demone remoto; la comunicazione può avvenire tramite socket o con delle API RESTful. L'utente non interagisce direttamente con il demone, bensì con il client, che rappresenta l'interfaccia utente primaria. Il fulcro del funzionamento di Docker è costituito dalle tre seguenti componenti:

- le image, che rappresentano un template di sola lettura usato per creare i container; Docker fornisce un sistema semplificato per creare nuove image o aggiornare quelle esistenti, oppure scaricare dalla rete quelle create da utenti terzi;
- i registri, archivi pubblici o privati che si occupano di mantenere le image, consentendone il caricamento o lo scaricamento;
- i container, intesi come contenitori di tutto ciò che è necessario affinché un'applicazione possa essere eseguita: ogni container rappresenta quindi una piattaforma applicativa isolata e sicura che può essere avviata, fermata, spostata o eliminata.

Ogni image è costituita da una serie di strati, i quali vengono combinati da Docker sfruttando degli union file system. Nel momento in cui vengono effetuate delle modifiche a una image, dei nuovi strati vengono creati e documentati, con il fine di descrivere dettagliatamente come ricreare tali modifiche; così, invece di dover sostituire un'intera image o ricostruirla da zero, solo tali strati vengono aggiunti o modificati. Questa strategia consente a Docker di gestire delle image ridotte in quanto a uso di risorse, poiché solo gli strati modificati necessitano di essere duplicati, diversamente da quanto avverrebbe ad esempio in una macchina virtuale. Ogni image può basarsi su una image base, sulla quale possono essere effettuate delle operazioni. In tal caso, Docker costruisce l'image partendo da quella base, per poi eseguire un insieme di istruzioni dettagliate che andranno a costituire un nuovo strato dell'image finale. Queste istruzioni sono memorizzate in un file chiamato Dockerfile. Un container è costituito da un sistema operativo, file e metadati. Ogni container viene costruito a partire da una image, la quale specifica cosa deve essere contenuto in esso, quale processo deve essere eseguito al suo avvio, e una varietà di altre informazioni di configurazione. Quando viene eseguito un nuovo container, Docker aggiunge un nuovo strato di lettura/scrittura in cima all'image, nel quale l'applicazione può essere eseguita.

Grazie a questa architettura, Docker rappresenta uno strumento in grado di contenere un'applicazione insieme alle sue dipendenze, garantendo flessibilità e portabilità relativamente alla sua esecuzione.

4.1.2 MongoDB

MongoDB [22] è un DBMS open-source orientato ai documenti. Rientra nella categoria dei DBMS NoSQL, infatti anziché utilizzare una struttura basata su tabelle come i database relazionali, MongoDB memorizza i dati in documenti JSON-like con schema dinamico (formato che prende il nome di BSON). Questa strategia mira ad avere migliori prestazioni rispetto ai RDBMS tradizionali, una disponibilità dei dati più elevata, e una scalabilità orizzontale su cluster che viene automatizzata grazie a un sistema chiamato *sharding*. I dati vengono organizzati in record chiamati *documenti*, ovvero strutture dati composte da coppie campo-valore. I valori associati ai campi possono contenere a loro volta altri documenti, array, o array di documenti. Questo annidamento di documenti e array consente di diminuire il numero di oggetti archiviati e di ovviare all'impossibilità di effettuare operazioni di join su documenti memorizzati in modo distribuito. Anziché usare tabelle, infatti, i documenti vengono organizzati in collezioni, tra le quali non è possibile effettuare operazioni di join in modo diretto. La robustezza del sistema viene incrementata da un sistema di ridondanza automatizzato. MongoDB presenta comunque degli svantaggi, infatti il suo sistema di indicizzazione dei dati determina un uso massiccio della memoria, e in quanto a spazio su disco, mediamente comporta un'occupazione maggiore rispetto ai DBMS relazionali.

4.2 Architettura del sistema

L'infrastruttura di supporto è stata progettata per essere installata sui computer che verranno utilizzati per le attività didattiche dei laboratori ALaDDIn, e deve consentire di avviare il software didattico associato all'attività in svolgimento, oltre a un'istanza MongoDB che servirà per archiviare i dati legati agli studenti partecipanti e agli esperimenti che essi compiono durante l'attività. Per l'esecuzione dei software didattici e di MongoDB, questa infrastruttura si avvale della piattaforma Docker. La piattaforma dispone di una image specifica per ognuna delle applicazioni legate ai software didattici e un'altra specifica per MongoDB (che d'ora in avanti verrà chiamata *image di supporto*). Queste image definiscono tutte le dipendenze software necessarie all'esecuzione dell'applicazione associata. Nel momento in cui è necessario avviare un determinato software, a partire dall'image relativa viene creato un container Docker in cui vengono avviate tutte le applicazioni specificate in essa. Nello specifico, per lo svolgimento dell'attività è previsto che venga avviato un container contenente il software didattico relativo, e un ulteriore

container specifico per MongoDB, in cui sarà presente un database destinato all'archiviazione dei dati tracciati durante tale attività. Il software didattico e l'istanza MongoDB sono quindi indipendenti tra loro, e il primo comunica all'altro i dati da memorizzare dopo che questi sono stati tracciati. Nel caso in cui si volesse integrare un nuovo software didattico in questa infrastruttura di supporto, essa è stata progettata in modo che sia necessario un quantitativo minimo di modifiche al software in questione al fine di adattarlo allo scopo.

Nell'architettura è prevista inoltre l'esistenza di una parte remota, su cui risiede un database centrale MongoDB con lo scopo di archiviare tutti i dati tracciati dai vari computer utilizzati nelle attività didattiche. I dati periferici saranno riversati su questo database periodicamente, sincronizzando da remoto i database MongoDB dei computer utilizzati nelle attività con quello centrale. In questo modo sarà possibile effettuare aggregazioni periodiche dei dati per poi procedere alla loro analisi.

Poiché questa infrastruttura non è destinata solamente ai computer appartenenti al laboratorio ALaDDIn, ma in prospettiva la si vuole rendere fruibile anche da docenti esterni per svolgere le attività didattiche nei loro istituti scolastici, ci si è occupati di adottare un sistema per installare la piattaforma Docker anche su sistemi operativi diversi da quelli Linux based. In particolare ci si è concentrati sul sistema operativo presumibilmente più diffuso per PC, ovvero Windows. Per essere compatibile anche con questo sistema infatti, Docker introduce delle differenze architetturali. Precisamente, è necessario sfruttare una macchina virtuale su cui risiederà l'host contenente il demone Docker. Il client risiederà invece sul sistema operativo Windows, e si interfaccerà con l'host risiedente nella macchina virtuale. Per lo sviluppo dell'infrastruttura ci si è avvalsi dell'applicazione Docker Toolbox, uno strumento che integra tutti i componenti necessari all'utilizzo di Docker nel sistema operativo citato (ne esiste anche una versione adatta ai sistemi Mac OS X). Poiché tutti i software didattici attuali di ALaDDIn, e presumibilmente anche quelli futuri, prevedono la presenza di un'interfaccia grafica, per fare in modo che questa possa essere visualizzata sul sistema Windows su cui risiede il client Docker, è necessario che sul sistema sia presente anche un server X, il quale dovrà ricevere gli output grafici relativi alle applicazioni eseguite nel container, e gestire gli input forniti dall'utente. Nello specifico, per lo sviluppo dell'infrastruttura è stata utilizzata l'applicazione MobaXterm [23]. Sul sistema Mac OS X questo passo non è necessario perché questi sistemi supportano X nativamente, ma bisogna comunque utilizzare una macchina virtuale.

Entrando nel dettaglio relativamente al tracciamento dei dati effettuato dall'infrastruttura, è stato progettato un sistema che prevede una fase di

registrazione da parte dei gruppi di studenti che utilizzano il software didattico, con conseguente generazione di un identificativo che consente di associare i singoli esperimenti allo specifico gruppo che li ha effettuati. Quindi il database MongoDB presente nell'infrastruttura deve contenere:

- una collezione relativa ai dati anagrafici degli studenti di ogni gruppo di lavoro;
- una collezione in cui vengono memorizzate le informazioni tracciate durante l'esecuzione dei vari esperimenti da parte degli studenti.

Infine, tenuto conto del fatto che la fase di registrazione dei gruppi di studenti che utilizzano il software (vedi paragrafo 4.3) è comune a tutte le attività, si è pensato di inserire il software relativo alla form di registrazione all'interno dell'image di supporto e di farlo eseguire prima dell'avvio del software didattico richiesto. La fase di registrazione termina con la generazione di un identificativo univoco per il gruppo di studenti. Questo identificativo farà parte di tutti i documenti MongoDB contenenti informazioni relative allo svolgimento delle attività effettuate da tale gruppo.

4.3 Risultato ottenuto

Una volta progettata l'architettura dell'infrastruttura, si è deciso di utilizzare un PC basato su Windows per verificare che gli obiettivi descritti nel paragrafo 4.2 fossero raggiungibili. Successivamente ci si è occupati della creazione dell'image di supporto.

4.3.1 Image di supporto

La definizione di questa image ha richiesto di posizionare in una cartella predefinita il corrispondente Dockerfile, due script che consentono l'avvio di alcune applicazioni necessarie all'interno del container e il software legato alla form di registrazione degli studenti. Quest'ultimo è stato sviluppato utilizzando HTML, Javascript e PHP per visualizzare in un browser una form che consente agli studenti del gruppo di lavoro di specificare per ognuno la propria età e il sesso. Una volta premuto un pulsante di conferma, nel database MongoDB viene inserito un documento contenente:

- un riferimento temporale relativo all'inserimento dei dati;
- un array contenente l'età e il sesso di ciascuno degli studenti del gruppo di lavoro.

Poiché ogni documento memorizzato in MongoDB è automaticamente associato a un identificatore univoco codificato tramite una stringa [1], si è scelto di utilizzare tale stringa per far riferimento al gruppo di lavoro. L'identificatore viene scritto all'interno di un file in una posizione fissa del file system (più avanti verrà fatto riferimento al percorso relativo a questa posizione indicandolo come "*/percorso_cartella*"). In questo modo, qualsiasi software didattico che viene eseguito successivamente è in grado di recuperare questa informazione e di utilizzarla per associare i dati degli studenti agli esperimenti che essi compiono. Siccome la registrazione è basata sull'uso di una form basata su HTML, è stato necessario integrare all'interno dell'image anche un server web. In particolare, ci si è avvalsi di Apache HTTP Server [24]. Analogamente, tutte le image relative ai software didattici della piattaforma devono integrare al loro interno anche un web browser, con cui si accederà alla pagina web contenente la form. La scelta è ricaduta su Mozilla Firefox [25].

Il Dockerfile legato alla creazione dell'image di supporto specifica le seguenti informazioni:

- la base dell'image corrisponde a quella contenente il sistema operativo Ubuntu, versione 14.04;
- vengono installati MongoDB e Apache HTTP Server, con le relative dipendenze;
- viene montato un volume che consente una memorizzazione persistente del database contenuto in MongoDB;
- vengono rese accessibili dall'esterno una porta per la comunicazione con MongoDB e una porta che consente l'accesso al server web;
- vengono copiati all'interno dell'image i file che descrivono la form di registrazione e due script che consentono l'avvio di MongoDB e di Apache;
- infine vengono eseguiti i due script appena citati.

La creazione dell'image avviene eseguendo all'interno della directory contenente il Dockerfile il seguente comando:

docker build -t mongo .

[1] L'univocità di questa stringa è altamente probabile.

In questo modo, sarà Docker a occuparsi di creare l'intera image, eventualmente scaricando dalla rete esterna le applicazioni necessarie specificate nel Dockerfile, se non sono presenti in locale.

Per creare e avviare un container relativo a questa image, è sufficiente eseguire in un terminale il seguente comando:

Comando 1 Creazione e avvio di un container per l'image di supporto

```
docker run --name mongo-server -p 81:80 -p 27017:27017
-v /var/lib/docker/mongodb_data:/data/db/
-v /percorso_cartella/shared_folder/config:/studID mongo
```

Questo comando specifica i seguenti elementi:

- il nome del container da creare e avviare;
- la rimappatura tra le porte aperte dal container e alcune porte del localhost (i container di Docker possono esporre delle porte, le quali possono essere collegate alle porte dell'host);
- i volumi da montare come cartelle condivise, che corrispondono ai percorsi relativi:
  - alla memorizzazione persistente del database;
  - al file contenente l'identificativo del gruppo di lavoro registrato;
- il nome dell'image da usare per la creazione del container.

4.3.2 Image dei software didattici

Dopo esserci occupati dell'image legata a MongoDB, ci si è occupati della creazione di una prima image legata a un software didattico. Precisamente, ci si è concentrati sul software associato al percorso didattico descritto in questa tesi. Grazie al modo in cui è stata progettata l'infrastruttura di supporto, l'adattamento e l'integrazione del software in questione hanno richiesto un quantitativo minimo di modifiche:

- nel file HTML principale è stato aggiunto uno script PHP che si occupa di avviare una sessione, leggere il file condiviso in cui è presente l'identificativo del gruppo registrato e memorizzare questo dato all'interno della sessione aperta;

- nel file Javascript che si occupa del funzionamento dell'elaboratore, si è fatto in modo che all'avvio di un nuovo esperimento venissero raccolti il numero relativo al livello di esplorazione corrente e la frase scelta per essere elaborata, per poi comunicare questi 2 dati a un server web attraverso una chiamata AJAX;

- questo server web è stato aggiunto al sistema al fine di eseguire il tracciamento delle attività degli studenti; infatti la chiamata AJAX è inviata a uno script che si occupa di recuperare l'identificativo precedentemente salvato nella sessione e di scrivere un documento contenente:

  - l'identificativo del gruppo di lavoro degli studenti;
  - un riferimento temporale relativo all'inserimento dei dati;
  - il livello di esplorazione e la frase scelta, passati come argomenti alla chiamata AJAX.

La descrizione di questa image è analoga a quella dell'image di supporto. In particolare, il Dockerfile specifica le seguenti informazioni:

- la base dell'image corrisponde a quella contenente il sistema operativo Ubuntu, versione 14.04;

- vengono installati Mozilla Firefox e Apache HTTP Server, con le relative dipendenze;

- vengono installate le dipendenze necessarie a X11, che servirà per la trasmissione dell'output grafico relativo all'applicazione Mozilla Firefox;

- viene resa accessibile dall'esterno una porta che consente l'accesso al server web;

- vengono copiati all'interno dell'image:

  - i file necessari alla fruizione del software didattico;
  - il file relativo alla scaletta del percorso didattico;
  - alcuni file che consentono di specificare a Firefox di includere tra i suoi segnalibri l'accesso al file con la scaletta;
  - uno script che consente l'avvio di Apache.

- infine vengono eseguiti tre comandi il cui scopo, nell'ordine, è di:

  - avviare Apache;

- avviare un'istanza di Firefox che visualizza la form di registrazione;
- avviare un'istanza di Firefox che accede al software didattico; questo comando viene eseguito solamente dopo che si è conclusa l'esecuzione di quelli precedenti: ciò consente al software didattico di avere a disposizione l'identificativo del gruppo di studenti che si è registrato.

La creazione dell'image avviene eseguendo all'interno della directory contenente il Dockerfile il seguente comando:

docker build -t ricorsione .

In questo modo, sarà Docker a occuparsi di creare l'intera image, eventualmente scaricando dalla rete esterna le applicazioni necessarie specificate nel Dockerfile, se non sono presenti in locale.

Per creare e avviare un container relativo a questa image, è sufficiente eseguire in un terminale il seguente comando:

Comando 2 Creazione e avvio di un container per il software didattico

```
docker run --rm --name containerRicorsione -p 80:80
-v /percorso_cartella/shared_folder:/shared_folder
-e DISPLAY=192.168.1.79:0.0 ricorsione
```

Questo comando specifica i seguenti elementi:

- il nome del container da creare e avviare;
- la rimappatura tra la porta aperta dal container e una porta del localhost;
- il volume da montare come cartella condivisa, che corrisponde al percorso del file contenente l'identificativo del gruppo di lavoro registrato;
- l'indirizzo di rete a cui è accessibile il server X;
- il nome dell'image da usare per la creazione del container.

Ipotizzando di lavorare con Windows e di avere installato Docker Toolbox, è necessario avviare un container relativo all'image di supporto e in seguito un secondo container relativo all'image con il software didattico, eseguendo sul terminale i comandi (1) e (2). In questo modo, grazie a MobaXTerm sarà aperta una finestra contenente un'istanza di Firefox che caricherà la pagina web relativa alla form di registrazione (vedi Figura 4.1).

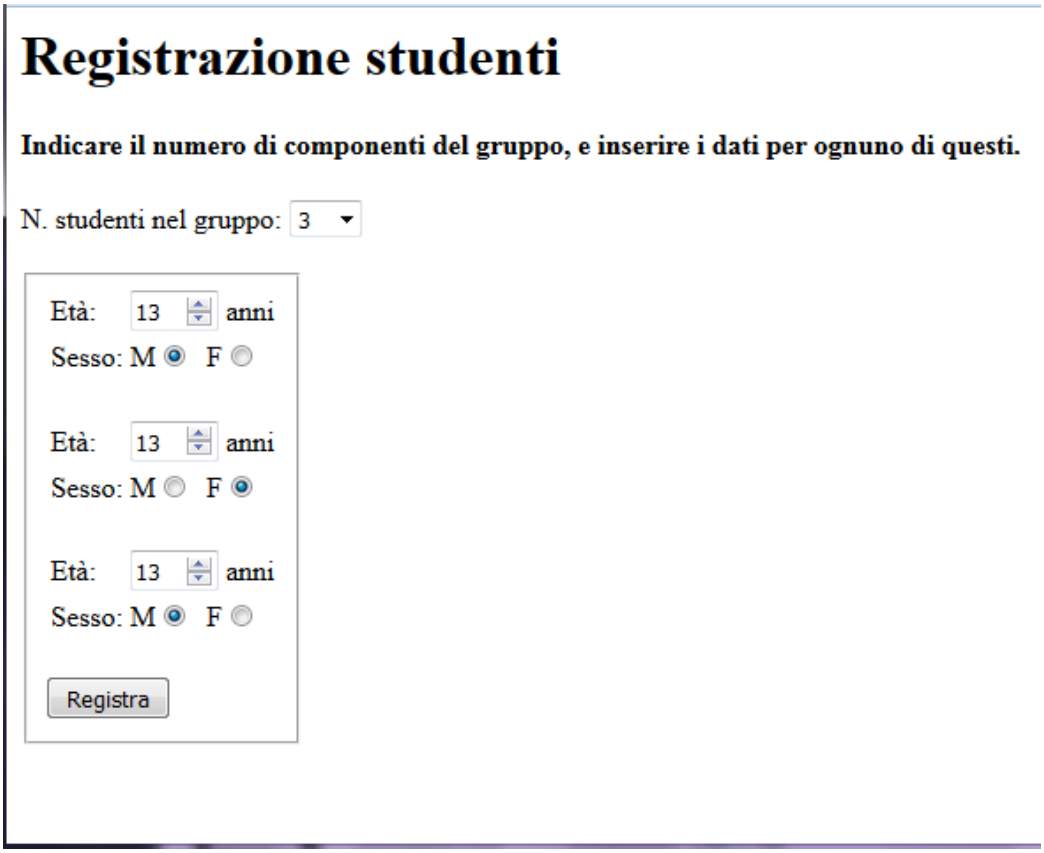


Figura 4.1: Screenshot della form di registrazione

4.3.3 Caso d'uso

Supponendo che sul computer stia lavorando un gruppo di due studenti, questi potranno specificare i loro dati anagrafici e registrarsi. Un esempio del documento inserito nella collezione del database relativa ai dati degli studenti, potrebbe essere il seguente:

{ "_id" : ObjectId("56f26614d462e60f008b4567"),
"timestamp" : ISODate("2016-03-23T09:47:00.397Z"),
"studenti" : [{ "eta" : 13, "sesso" : "M" },
{ "eta" : 13, "sesso" : "F" }]}

A registrazione compiuta, la form mostrerà un avviso di conferma che suggerirà di chiudere la finestra e attendere l'avvio del software didattico. L'identificativo del gruppo di studenti corrisponde all'id di tale documento, che in tal caso è *"56f26614d462e60f008b4567"*, il quale sarà scritto in un file condi-

viso. Una volta che gli studenti avranno chiuso la schermata, sarà eseguito il comando che apre una nuova finestra contenente un'istanza di Firefox che caricherà il software didattico. Supponiamo che gli studenti effettuino un esperimento con il software nel primo livello di esplorazione, facendo elaborare una delle frasi preimpostate del software. In tal caso, un esempio del documento inserito nella collezione del database relativa agli esperimenti effettuati, potrebbe essere il seguente:

{"studID" : "56f26614d462e60f008b4567",
"timestamp" : ISODate("2016-03-23T09:47:00.397Z"),
"livello" : 1, "stringaInput" : "amenic la omaidna"}

Notare che la stringa contenuta nel campo *studID* corrisponde esattamente all'identificativo che è stato scritto precedentemente nel file condiviso. Per ogni altro esperimento che sarà fatto con il software, sarà inserito nel database un nuovo documento con tale forma.

4.4 Integrazione di altri software

Una volta sviluppata l'infrastruttura e verificato che questa consentisse di raggiungere gli obiettivi prefissati, si è voluto adattare e integrare in essa anche un ulteriore software didattico facente parte dei laboratori ALaDDIn, in modo da verificare che l'infrastruttura presentasse davvero la flessibilità auspicata. Il software preso in considerazione per tale scopo è quello associato al percorso didattico *Clickomania* [26], in cui ai partecipanti è chiesto di implementare il noto videogioco omonimo usando l'ambiente di programmazione Blockly [7], con l'obiettivo di farli focalizzare sul concetto di struttura dati.

4.4.1 Descrizione di Clickomania

Il gioco *Clickomania* è un solitario, noto anche con il nome di *Chain Shot!* oppure *Same Game*. Il campo di gioco è costituito da una parete quadrata inizialmente coperta da 81 mattoni, uno per casella, di colori diversi, e l'obiettivo del gioco è rimuovere dalla parete tutti i mattoni, selezionando via via aree di mattoni adiacenti dello stesso colore. A ogni mossa il giocatore clicca su un mattone e può dare inizio a un'infezione che contagia i mattoni vicini. La cosiddetta *area di contagio* è definita come segue: se il mattone selezionato non ha nessun vicino (sopra, sotto, a destra o a sinistra) dello stesso colore, l'area di contagio è vuota; se invece il mattone selezionato ha

uno o più vicini dello stesso colore, allora l'area di contagio comprende il mattone cliccato e tutti i mattoni collegati a esso che possiedono lo stesso colore (vedi Figura 4.2).

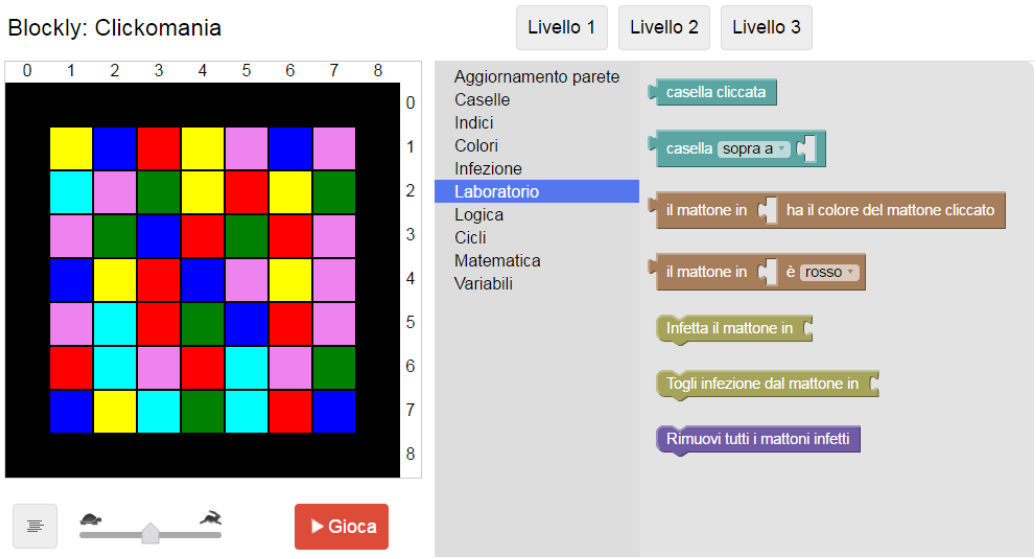
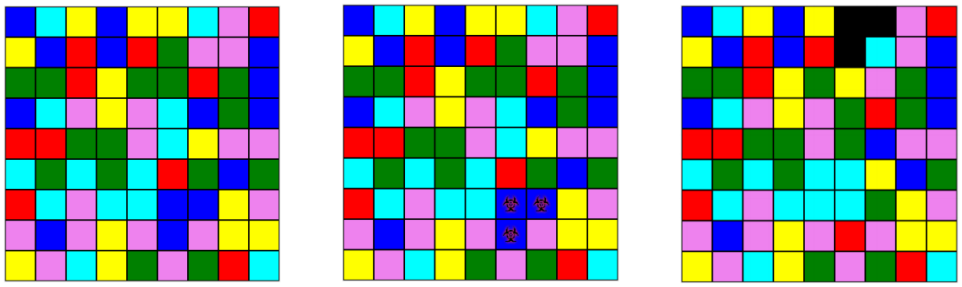


Figura 4.2: Screenshot della pagina web

Una semplificazione del gioco consiste nel considerare, invece di tutti i mattoni vicini, solo i mattoni sopra o sotto la casella cliccata; in questo caso si definisce *striscia di contagio* la striscia verticale formata dal mattone cliccato e dai suoi vicini dello stesso colore. In questa semplificazione, l'infezione contagia dunque i mattoni della striscia di contagio del mattone cliccato, invece di quelli dell'area di contagio. Tutti i mattoni infetti vengono quindi rimossi e il campo di gioco si ricompatta: in ogni colonna, le caselle lasciate libere dai mattoni rimossi vengono riempite dai mattoni che si trovano sopra, i quali cadono verso il basso. Il gioco finisce quando ogni mattone rimasto non ha mattoni vicini dello stesso colore.

Argomento del laboratorio è la realizzazione di un programma per giocare alla versione semplificata di Clickomania, o più precisamente, un programma che aggiorni la parete di gioco in seguito a ogni mossa del giocatore, utilizzando l'ambiente di programmazione Blockly. Tale ambiente consente di costruire dei programmi a partire da blocchi colorati predefiniti, che si possono incastrare come in un puzzle, in cui ogni blocco corrisponde a una specifica istruzione oppure a una struttura di controllo. Non esiste un solo programma che permette di giocare a Clickomania: i blocchi a disposizione possono essere combinati a formare tanti diversi programmi, tutti funzionanti. Il laboratorio è suddiviso in tre fasi in cui si prevede il raggiungimento dei

seguenti obiettivi. Inizialmente si richiede l'implementazione di un programma che, in seguito al clic su un mattone della parete, provoca il contagio e la rimozione di una striscia composta da due mattoni vicini dello stesso colore: quello cliccato e quello adiacente. In seguito, il programma implementato dovrà provocare il contagio e la rimozione di una striscia di contagio che si estende in una sola direzione a partire dal mattone cliccato. Infine, il programma implementato dovrà provocare il contagio e la rimozione di una striscia di contagio che si estende in entrambe le direzioni a partire dal mattone cliccato. Una volta costruito un programma, si può iniziare il gioco cliccando sul bottone "Gioca". A questo punto, ad ogni clic del giocatore la parete verrà aggiornata in base alle istruzioni del programma costruito (cfr. Figura 4.3).



(a) (b) (c)

Figura 4.3: (a) raffigura la configurazione iniziale della parete, (b) raffigura l'infezione conseguente al clic sul mattone in riga 6 e colonna 5, (c) raffigura la parete dopo la rimozione dei mattoni infetti

Anche questo software quindi consente agli studenti di effettuare degli esperimenti, in particolare, dopo aver sviluppato un programma che determina il funzionamento del gioco, essi possono testarlo sulle griglie che vengono generate casualmente dall'ambiente. Infatti, integrando questo software all'interno dell'infrastruttura di supporto sviluppata, si è pensato di tracciare tali esperimenti, archiviando in particolare i blocchi che costituiscono il programma ideato e la griglia di gioco che viene generata nel momento in cui viene avviato il gioco (ovvero il testing del programma). Inoltre, il software presenta 3 diversi livelli di gioco selezionabili dall'utente, e sono presenti anche alcuni slot in cui gli studenti possono salvare le loro soluzioni. Nel momento in cui uno studente preme il pulsante "Gioca", oppure memorizza il proprio programma in uno degli slot disponibili, i dati relativi vengono inseriti in un database MySQL remoto [27], in modo da mantenere uno storico delle soluzioni salvate e consentendo anche agli studenti di riottenere tali soluzioni nel caso in cui accedessero al software in seguito. Per questo motivo,

tra le informazioni che si intendeva tracciare nell'infrastruttura di supporto, si è deciso di considerare anche i dati relativi al livello corrente e allo slot selezionato.

4.4.2 Adattamenti effettuati

Il software del laboratorio *Clickomania* presenta delle caratteristiche simili al software didattico a cui ci si riferisce nel paragrafo 4.3, inoltre è anch'esso sviluppato usando un linguaggio eseguito all'interno di un browser, quindi per riuscire ad adattarlo e integrarlo nell'infrastruttura di supporto sono state effettuate in buona parte le stesse operazioni descritte precedentemente. Risulta pertanto sufficiente illustrare le operazioni aggiuntive.

Innanzitutto è stato necessario apportare alcune modifiche ai file sorgenti del software, in particolare:

- si è fatto in modo che alla pressione del pulsante "Gioca" o di uno degli slot per salvare le soluzioni, venissero raccolti precisamente questi dati:
  - l'identificativo del gruppo di lavoro degli studenti;
  - un riferimento temporale relativo all'inserimento dei dati;
  - l'insieme dei blocchi che costituiscono il programma definito dagli studenti, che viene trattato dal software come una stringa con sintassi XML;
  - la griglia di gioco generata al momento in cui viene avviato il gioco, che viene trattata come un array di interi, dove ogni intero va a rappresentare il colore di una specifica casella della griglia;
  - il livello di gioco selezionato, che viene rappresentato con un valore da 0 a 2;
  - lo slot selezionato per il salvataggio della soluzione, che viene rappresentato con un valore da 0 a 4 (nel caso in cui fosse stato premuto il pulsante "Gioca", avremmo settato '0' come valore di default);
- gli script PHP che si occupano di leggere o scrivere i dati nel database MySQL sono stati modificati in modo da effettuare tali operazioni nel database MongoDB reso accessibile dal container dell'infrastruttura, facendo riferimento a un documento che contiene i dati sopra citati.

Per la creazione dell'image contenente tale software, la cartella di creazione deve contenere gli stessi elementi previsti per l'image descritta nel paragrafo 4.3, a eccezione dei file sorgenti riguardanti il software specifico e del

file relativo alla scaletta del percorso didattico. Anche il Dockerfile contenuto in tale cartella definisce pressoché le stesse operazioni previste per l'altra image, a eccezione del comando che avvia l'istanza di Firefox per l'accesso al software didattico, che in questo caso accederà alle pagine web relative al software Clickomania. La creazione dell'image avviene eseguendo all'interno della directory contenente il Dockerfile il seguente comando:

docker build -t clickomania .

Per creare e avviare un container relativo a questa image, è sufficiente eseguire in un terminale il seguente comando:

Comando 3 Creazione e avvio di un container per Clickomania

```
docker run --rm --name containerClicko -p 80:80
-v /percorso_cartella/shared_folder:/shared_folder
-e DISPLAY=192.168.1.79:0.0 clickomania
```

4.4.3 Caso d'uso

Supponiamo che gli studenti, dopo aver avviato il software, effettuino un esperimento che li porta a ottenere la configurazione di gioco descritta in Figura 4.4.

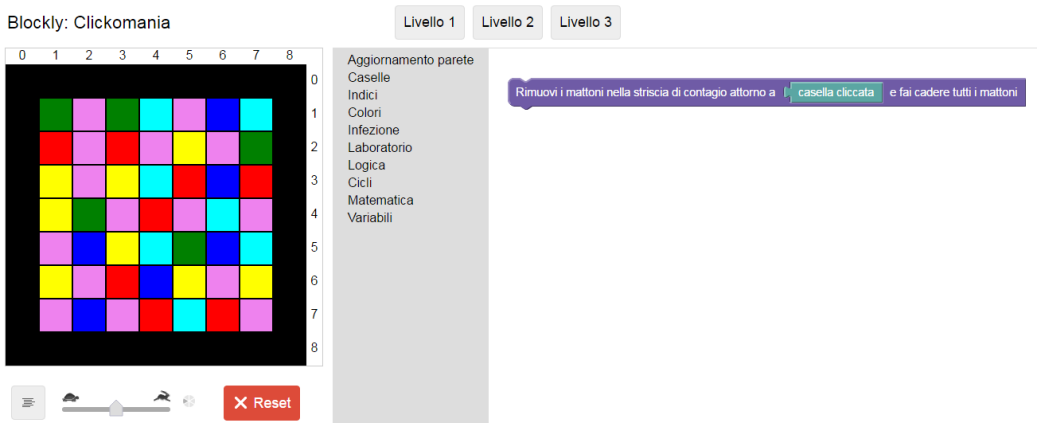


Figura 4.4: Screenshot della configurazione di esempio

Un esempio del documento inserito nella collezione del database relativa agli esperimenti effettuati con Clickomania, potrebbe essere il seguente:

{ "_id" : ObjectId("56f293bdd9a3f413008b4567"),
"studID" : "56f2661b41d54d0f008b4567", "livello" : 0, "slot" : 0,
"xml" : "<xml xmlns="http://www.w3.org/1999/xhtml">
<block type="solutionStrip" id="37" inline="true" x="8" y="10">
<value name="P"><block type="getClickPosition" id="44">
</block></value></block></xml>",
"griglia":"[[9,9,9,9,9,9,9,9,9],[9,6,4,6,5,4,1,5,9],[9,2,4,2,4,3,4,6,9],
[9,3,4,3,5,2,1,2,9],[9,3,6,4,2,4,5,4,9],[9,4,1,3,5,6,1,5,9],
[9,3,4,2,1,3,4,3,9],[9,4,1,4,2,5,2,4,9],[9,9,9,9,9,9,9,9,9]]",
"timestamp" : ISODate("2016-03-23T13:01:49.845Z") }

Grazie a questa ulteriore prova si è riusciti a dimostrare come per l'integrazione di un software didattico all'interno dell'infrastruttura siano necessari pochi sforzi, raggiungendo gli obiettivi prefissati.

Conclusioni e sviluppi futuri

In questa tesi è stato descritto il lavoro di progettazione e messa a punto di un percorso didattico che affronta la ricorsione come tecnica per il *problem solving*. Si è effettuata una prima sperimentazione di tale percorso sul campo in una classe terza di una scuola secondaria di primo grado. In base alle analisi effettuate sui risultati ottenuti, il percorso ha dimostrato una certa efficacia nel raggiungimento degli obiettivi che ci eravamo prefissati. In particolare si ritiene che gli studenti che hanno partecipato alla sperimentazione abbiano acquisito la conoscenza delle caratteristiche principali della ricorsione e del processo messo in atto durante l'esecuzione di alcuni esempi di algoritmi ricorsivi, avendo consapevolezza di cosa avviene nei singoli passi di questo processo, ma anche avendo una visione di insieme di come questi passi sono collegati tra loro.

Una singola sperimentazione però non è sufficiente per trarre conclusioni pienamente attendibili; sarebbe necessario effettuarne altre, così da avere una quantità più ampia di dati e di osservazioni da analizzare.

A partire dalle analisi fatte sulla sperimentazione effettuata, si è previsto di apportare alcune modifiche alla prima fase del percorso didattico, con lo scopo di rendere più fluida l'esecuzione dell'algoritmo ricorsivo da parte degli studenti. Sono previsti miglioramenti anche per i materiali usati nella seconda fase, ovvero lo strumento software e la scheda che ne guida l'utilizzo, allo scopo di rendere più chiari:

- l'interfaccia grafica della pagina web;
- lo svolgimento del processo relativo all'esecuzione dell'algoritmo;
- il contesto a cui si riferiscono le domande della scheda.

Sono stati inoltre ipotizzati due possibili ampliamenti del percorso basati su: un'attività di progettazione di un algoritmo di ricerca (*ricerca binaria*) e un'attività di analisi di un noto algoritmo, il Merge Sort, più complesso e adatto per studenti con una più elevata capacità di astrazione.

Per quanto riguarda l'infrastruttura software progettata con lo scopo di supportare l'utilizzo dei software didattici, si è riusciti a dimostrare che questa presenta la flessibilità auspicata, in quanto per integrare al suo interno un nuovo software didattico sono necessari pochi sforzi.

Per facilitare l'utilizzo di tale infrastruttura, si è previsto lo sviluppo di un'interfaccia grafica che consenta di scegliere in modo più pratico il software didattico che si intende avviare. Inoltre si è previsto di:

- sviluppare un sistema che semplifichi l'invio dei dati archiviati a un database MongoDB su server remoto;
- sviluppare un frontend dedicato al laboratorio ALaDDIn che consenta un accesso facilitato ai dati archiviati dall'infrastruttura.

Bibliografia

[1] T.H. Cormen. *Introduction to Algorithms*. MIT Press, 2009.

[2] Jeff Erickson. Algorithms. `http://jeffe.cs.illinois.edu/teaching/algorithms/all-recursion.pdf`, 2015.

[3] O. Hazzan, T. Lapidot, and N. Ragonis. *Guide to Teaching Computer Science: An Activity-Based Approach*. Springer London, 2011.

[4] Maciej M. Syslo and Anna Beata Kwiatkowska. *Informatics in Schools. Teaching and Learning Perspectives: 7th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2014, Istanbul, Turkey, September 22-25, 2014. Proceedings*, chapter Introducing Students to Recursion: A Multi-facet and Multi-tool Approach, pages 124–137. Springer International Publishing, Cham, 2014.

[5] Giovanni Marcianò. Usare il linguaggio logo per costruire micromondi. `http://www.robocupjr.it/margi/pubblicazioni/1203.pdf`.

[6] T.W. Gray and J. Glynn. *Exploring Mathematics with Mathematica: Dialogs Concerning Computers and Mathematics*. Number v. 1 in Exploring Mathematics with Mathematica: Dialogs Concerning Computers and Mathematics. Addison-Wesley, 1991.

[7] Sito ufficiale di Blockly. `https://developers.google.com/blockly`.

[8] Sito ufficiale di Scratch. `https://scratch.mit.edu`.

[9] Sito di Recursive Drawing. `http://recursivedrawing.com`.

[10] Violetta Lonati, Dario Malchiodi, Mattia Monga, and Anna Morpurgo. Is coding the way to go? In Andrej Brodnik and Jan Vahrenhold, editors, *8th international conference on informatics in schools: situation, evolution, and perspective*, volume 9378 of *LNCS*, pages 165–174, Switzerland, 2015. Springer International Publishing.

[11] Robert Harper. It is what it is (and nothing else). https://existentialtype.wordpress.com/2016/02/22/it-is-what-it-is-and-nothing-else, 2016.

[12] Url del canale YouTube di AlgoRythmics. https://www.youtube.com/user/AlgoRythmics.

[13] L. Vygotsky. *Mind in Society: Development of Higher Psychological Processes*. Cambridge: Harvard University Press, 1978.

[14] Sito ufficiale di Aladdin. http://aladdin.unimi.it/index.html.

[15] Carlo Bellettini, Violetta Lonati, Dario Malchiodi, Mattia Monga, Anna Morpurgo, Mauro Torelli, and Luisa Zecca. Extracurricular activities for improving the perception of informatics in secondary schools. In Yasemin Gülbahar and Erinc Karatas, editors, *Informatics in schools. teaching and learning perspectives*, volume 8730 of *Lecture Notes in Computer Science*, pages 161–172. Springer International Publishing, 2014.

[16] Mordechai Ben-Ari. Constructivism in computer science education. *ACM SIGCSE Bulletin*, volume 8, 1998.

[17] André Giordan. Le modèle allosterique et les theories contemporaines. *In Laboratoire de didactique et d'epistemologie des sciences*, 2012.

[18] Carlo Bellettini, Violetta Lonati, Dario Malchiodi, Mattia Monga, Anna Morpurgo, and Mauro Torelli. What you see is what you have in mind: constructing mental models for formatted text processing. In *Proceedings of ISSEP 2013*, number 6 in Commentarii informaticae didacticae, pages 139–147. Universitätsverlag Potsdam, 2013.

[19] Sito ufficiale di Code.org. https://code.org.

[20] Sito ufficiale della libreria Raphael. http://dmitrybaranovskiy.github.io/raphael.

[21] Sito ufficiale della piattaforma Docker. https://www.docker.com.

[22] Sito ufficiale del DBMS MongoDB. https://www.mongodb.org.

[23] Sito ufficiale di Mobaxterm. http://mobaxterm.mobatek.net.

[24] Sito ufficiale del server web Apache. https://httpd.apache.org.

[25] Sito ufficiale del browser web Mozilla. https://www.mozilla.org.

[26] Sito dell'applicazione Clickomania. http://click-aladdinunimi.rhcloud.com/apps/kangourou.

[27] Sito ufficiale del DBMS MySQL. https://www.mysql.it.

Appendice - Scheda di lavoro

Scheda - L'elaboratore misterioso

Lo scienziato Albert ha costruito un elaboratore particolare che gli consente di risolvere un certo tipo di problemi. Per la precisione, questo elaboratore esegue delle operazioni sulle frasi: inserendo una ***frase da elaborare***, si ottiene una ***frase elaborata***. Purtroppo però, Albert ha qualche problemino di memoria, e dato che ha costruito l'elaboratore tanto tempo fa, non riesce a ricordare cosa fa esattamente, né come funziona di preciso. Siete in grado di aiutarlo a rinfrescarsi la memoria?

Provate a fare degli esperimenti. Il programma mostrerà i pulsanti ***Livello 1***, ***Livello 2*** e ***Livello 3***. Attraverso queste tre diverse ***modalità di elaborazione*** potrete esplorare sempre più in profondità il "meccanismo magico" con cui funziona l'elaboratore. Scegliete e modificate le ***frasi da elaborare***. Nei livelli **2** e **3** comparirà il pulsante ***Esplora***: cliccandolo si fermerà il tempo e potrete vedere alcuni **indizi**.

Per iniziare a giocare, premete ***Livello 1***. Fate qualche esperimento, e quando siete pronti rispondete alle seguenti domande.

Domande:

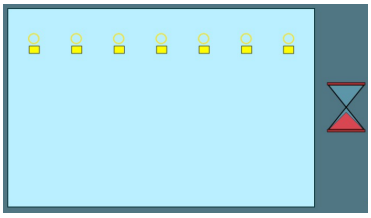
1. La *frase da elaborare* deve essere lunga almeno 15 lettere. Facciamo finta che si possa inserire una frase più corta. Se la frase fosse *"CIAO MAMMA"*, quale *frase elaborata* si otterrebbe?

Livello 2:

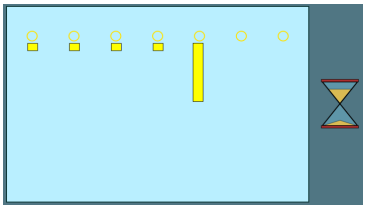
2. Osserva la torre gialla mentre scorre sotto la fila di cerchiolini. Come cambia la torre quando la clessidra è gialla? E come cambia quando la clessidra è arancione?

3. Nel *Livello 2*, cosa rappresenta il numero blu che compare quando clicchiamo su *Esplora*?

4. Immaginiamo di poter elaborare la frase *"GIOCARE"* e di premere *Livello 2*. Che lettere conterrebbe la torre gialla cliccando su *Esplora* in questo momento dell'elaborazione?
Nota: la clessidra è **rossa**.



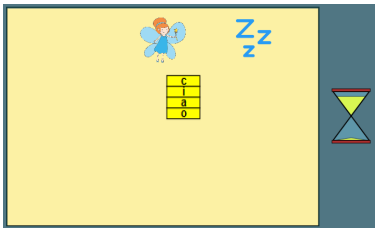
5. Immaginiamo ancora di elaborare la frase *"GIOCARE"* e di premere *Livello 2*. Che lettere conterrebbe la torre gialla cliccando su *Esplora* in questo momento?
Nota: la clessidra è **arancione**.



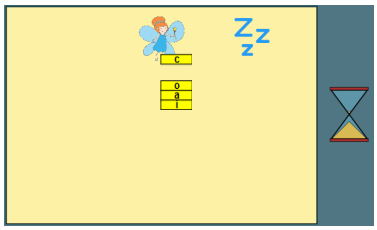
Livello 3:

6. Nel *Livello 3*, che tipo di operazioni sono capaci di svolgere le **fatine**?

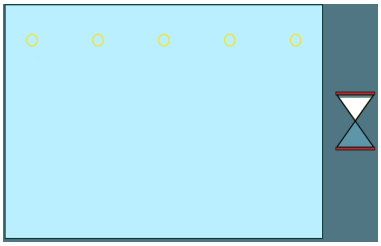
7. Immaginiamo di essere nel *Livello 3*, e che la torre gialla sia **appena arrivata** nella posizione indicata dall'immagine seguente. Che lettere conterrà la torre **dopo** l'operazione della fatina?
Nota: la clessidra è **gialla**.



8. E invece in quest'altro caso?
Nota: la clessidra ora è **arancione**!



9. Immaginiamo di poter elaborare la frase *"CORSA"* e di premere *Livello 3*. Se io volessi riuscire a vedere una fatina che **attacca** la lettera *'R'* alla torre gialla, su quale cerchiolino dovrei cliccare? E in che posizione dovrebbe essere la torre? Segnate una X sul cerchiolino che dovremmo cliccare, e disegnate la torre nella posizione giusta. Scrivete anche il colore che dovrebbe avere la clessidra.



Ringraziamenti

Desidero ringraziare la Prof.ssa Lonati, la Prof.ssa Morpurgo e il Prof. Malchiodi per avermi dato la possibilità di avvicinarmi al mondo della didattica e per avermi supportato durante la realizzazione del lavoro descritto in questa tesi, soprattutto per quanto riguarda il sostegno morale datomi nell'ultimo periodo di lavoro, che mi è stato di immenso aiuto.

Ringrazio anche il Prof. Monga, che ha collaborato di buon grado e con una certa costanza durante le fasi di progettazione, sia del percorso didattico, sia dell'infrastruttura software.

Un ringraziamento va anche alla Prof.ssa Martina Palazzolo, che ci ha concesso di sperimentare il percorso didattico con la sua classe scolastica, mostrando sempre un grande interesse per il lavoro che stavamo svolgendo.

Infine voglio ringraziare la mia famiglia, che mi ha sostenuto durante tutto il percorso di studi universitari, e tutti gli amici che mi sono stati vicini nei momenti di gioia, ma anche di difficoltà, che si sono presentati durante questo percorso.