Alma Mater Studiorum · Università di Bologna

facoltà di scienze matematiche, fisiche e naturali

Corso di Laurea Specialistica in Informatica

Integrazione di Java e Mathematica in un software per la didattica della geometria nella scuola primaria

Tesi di Laurea in Matematica Computazionale

Relatore:
Chiar.mo Prof.
Giulia Spaletta

Presentata da:
Laura Nicli

III Sessione
Anno Accademico 2009/10

Parole chiave

- Geometria nella scuola primaria
- Software didattico
- Integrazione di Java e Mathematica

Indice

Elenco delle figure

Introduzione

In questa tesi verrà illustrata un'applicazione che sfrutta l'integrazione di Java e Mathematica, per lo sviluppo di un software di ausilio alla didattica della geometria nella scuola primaria.

Questo progetto nasce dalla necessità, da parte degli insegnanti, di disporre di uno strumento che permetta loro di spiegare al meglio come avviene il calcolo dell'area di vari poligoni. Questa necessità prevede che il software sia portabile, per far fronte alla diversità delle macchine su cui verrà utilizzato, e che sia estremamente semplice da utilizzare, in quanto gli utenti saranno persone a cui non deve essere richiesta alcuna particolare conoscenza informatica.

Lo strumento sviluppato ha, quindi, come obiettivo, quello di aiutare gli alunni a comprendere a fondo quali sono le motivazioni che stanno dietro a tutte le formule, solitamente utilizzate, per il calcolo dell'area dei poligoni. L'applicazione ha il compito di accompagnare il bambino in un percorso dinamico ed interattivo, caratterizzato dall'utilizzo di animazioni ed immagini in movimento che, graficamente, operano delle trasformazioni al poligono

in esame, fino a rendere chiara la spiegazione dell'insegnante, riguardo alla formula usata per il calcolo dell'area del poligono stesso.

La peculiarità del software, sviluppato in questo lavoro di tesi, è l'integrazione operata tra il linguaggio di programmazione Java e Mathematica, un ambiente di calcolo simbolico e numerico dalle infinite capacità. Java è stato scelto per la sua indipendenza dalla piattaforma, mentre Mathematica è stato scelto per la sua potenza e facilità di creazione e gestione di immagini graficamente manipolabili.

Nei prossimi capitoli verranno descritte a fondo le caratteristiche di quest'applicazione, i dettagli implementativi ritenuti rilevanti e gli accorgimenti adottati per fare in modo di soddisfare al meglio le richieste degli utenti finali.

Nel capitolo 1, verrà illustrato il contesto per il quale è stata sviluppata questa applicazione, la scuola primaria, descrivendone l'importantissimo ruolo che ricopre nell'educazione di un bambino; verrà data particolare attenzione alla descrizione dei compiti, sempre più rilevanti, della tecnologia come ausilio della didattica, soprattutto della geometria.

Nel capitolo 2, verranno descritti in dettaglio gli strumenti utilizzati per sviluppare questa applicazione, con particolare attenzione ai moduli sfruttati per l'integrazione di Java e Mathematica.

Nel capitolo 3, sarà illustrata, infine, l'implementazione effettiva del progetto, attraverso la descrizione di tutte le sue componenti computazionali.

Al termine di questo documento di tesi è poi presente un'appendice, che contiene la descrizione, attraverso *screen-shots*, di un esempio di esecuzione

del programma, per meglio illustrare come avviene la spiegazione del calcolo dell'area di una figura geometrica.

Capitolo 1

La geometria nella scuola primaria

1.1 Gli obiettivi della scuola primaria

L'apprendimento scolastico, oggi, è solo una delle tante esperienze di formazione che i bambini vivono; per acquisire le competenze specifiche spesso non vi è bisogno dei contesti scolastici, ma proprio per questo la Scuola Primaria è chiamata a garantire a tutti gli alunni le condizioni culturali, relazionali, didattiche ed organizzative idonee a "rimuovere gli ostacoli di ordine economico e sociale" che, limitando di fatto la libertà e la giustizia verso i cittadini, "impediscono il pieno sviluppo della persona umana", indipendentemente dal sesso, dalla razza, dalla lingua, dalla religione, dalle opinioni politiche e dalle condizioni personali e sociali (art. 3 della Costituzione italiana).

In continuità con la famiglia, la Scuola Primaria insegna a tutti i bambini l'alfabeto dell'integrazione affettiva della personalità e pone le basi per un'immagine realistica, ma positiva, di sé, in grado di valorizzare come potenzialità personale anche ciò che, in determinati contesti di vita, può apparire e magari è un'oggettiva limitazione.

Per queste ragioni, la Scuola Primaria è l'ambiente educativo di apprendimento, nel quale ogni bambino trova le occasioni per maturare progressivamente le proprie capacità di autonomia, di azione diretta, di relazioni umane, di progettazione e verifica, di esplorazione, di riflessione logico-critica e di studio individuale; è il luogo in cui ci si abitua a radicare le conoscenze (sapere) sulle esperienze (il fare e l'agire), promuove negli alunni l'acquisizione di tutti i tipi di linguaggio e un primo livello di padronanza delle conoscenze e delle abilità, comprese quelle metodologiche di indagine, aiutando il passaggio dal "sapere comune" al "sapere scientifico"; ciò al fine di poter affrontare positivamente l'incertezza e la mutevolezza degli scenari presenti e futuri. I bambini che entrano nella Scuola Primaria hanno già maturato una loro ingenua, ma non per questo meno unitaria, organica e significativa visione del mondo e della vita.

Le finalità della scuola devono essere definite a partire dalla persona che apprende, con l'originalità del percorso individuale e le aperture offerte dalla rete di relazioni che la legano alla famiglia e agli ambiti sociali. La Scuola Primaria si propone, anzitutto, di apprezzare questo patrimonio conoscitivo, valoriale e comportamentale ereditato dal bambino, e di dedicare particolare

attenzione alla sua considerazione, esplorazione e discussione comune, partendo dalla valorizzazione dell'espressione corporea, condizione e risultato di tutte le altre dimensioni della persona: razionale, estetica, sociale, operativa, affettiva e morale.

Le strategie educative e didattiche tengono conto della singolarità e complessità di ogni persona, della sua articolata identità, delle sue aspirazioni, capacità e delle sue fragilità, nelle varie fasi di sviluppo e di formazione; per questo, mireranno ad offrire agli studenti occasioni di apprendimento dei saperi e dei linguaggi culturali di base con specifiche finalità.

- Acquisire gli strumenti di pensiero necessari per apprendere e selezionare le informazioni.

- Elaborare metodi e categorie che siano in grado di fare da bussola negli itinerari personali.

- Stimolare l'autonomia di pensiero degli studenti, orientando la propria didattica alla costruzione di saperi, a partire da concreti bisogni formativi.

In conclusione, il percorso complessivamente realizzato nella Scuola Primaria promuove l'educazione integrale della personalità degli alunni, stimolandoli all'autoregolazione degli apprendimenti, ad un'elevata percezione di autoefficacia, all'autorinforzo cognitivo e di personalità, alla massima attivazione delle risorse di cui sono dotati, attraverso l'esercizio dell'autonomia persona-

le, della responsabilità intellettuale, morale e sociale, della creatività e del gusto estetico [IND].

1.2 Tecnologia educativa

> *"Le tecnologie didattiche riguardano la definizione e lo sviluppo di modelli teorici e la messa a punto di metodologie e di sistemi tecnologici per risolvere problemi riguardanti l'apprendimento umano in situazioni finalizzate e controllate [...] Ciò che caratterizza le tecnologie didattiche è l'approccio sistematico ed interdisciplinare che, mutuando conoscenze da settori differenti (psicologia cognitiva, informatica, pedagogia, comunicazioni ecc..) le integra in un sistema complesso, controllato e finalizzato al raggiungimento di specifici obiettivi formativi"* [MID]

Definire la tecnologia educativa non è facile, benché numerosi siano i tentativi presenti nella letteratura. Il punto di partenza è una definizione insieme operativa e intuitiva: si opera e si ricerca nell'ambito della tecnologia educativa tutte le volte in cui, in riferimento ad ipotesi sui processi (naturali e non) che influenzano gli apprendimenti, si cerca di migliorare le condizioni per favorirli, avvalendosi di particolari metodologie, sovente basate sull'impiego di mezzi tecnologici adeguatamente utilizzati.

Numerosi sono i modi e le circostanze in cui, nel corso del tempo od in riferimento a diverse teorie, si è sviluppata la tecnologia educativa. Diversi

sono i programmi di ricerca ed i filoni avviati; qui di seguito ne vengono elencati alcuni.

- Supporto all'istruzione tradizionale (uso di lavagna luminosa, slide-show, audiovisivi, a supporto della lezione cattedratica).
- Costruzione di ambienti d'apprendimento (gioco, simulazione al computer, simulazione come costruzione di un contesto sociale e così via).
- Analisi delle trasformazioni indotte dai cosiddetti "nuovi media" nei processi di lettura/scrittura.
- Strumenti per "adeguare" processi di lettura/scrittura alle caratteristiche cognitive umane (ipertesti, ipermedia e così via).

Ovviamente ognuno di questi filoni d'indagine teorica ed operativa poggia su teorie sviluppate nel campo sovraordinato delle Scienze dell'Educazione e in particolare sulle Teorie dell'Apprendimento.

Riferirsi alla tecnologia educativa ed ai suoi modelli apre altre prospettive. Serve, ad esempio, a chiarire che il compito della ricerca di introdurre Tecnologie della Comunicazione e dell'Informazione nella scuola, non è banalmente "adeguare la scuola al mondo che cambia", ma, al contrario, quello di "piegare" le tecnologie al suo fine principale: la qualità degli apprendimenti degli studenti.

Dunque, non una scuola che insegue il mondo per adeguarsi, ma una scuola che, con consapevolezza e capacità di analisi, impiega, con competenza

teorica e metodologica, gli artefatti ed i concetti che emergono dal mondo tecnologico, adattandoli alle proprie necessità.

1.3 Gli strumenti a disposizione - LIM

Dal 14 gennaio 2009 il Ministero dell'Istruzione dell'Università e della Ricerca promuove il Piano per l'Innovazione Digitale nella Scuola, per sviluppare e potenziare l'innovazione didattica attraverso l'uso della lavagna interattiva, strumento che favorisce l'integrazione delle Tecnologie dell'Informazione e della Comunicazione nei processi di apprendimento in classe.

La Lavagna Interattiva Multimediale (LIM), o lavagna digitale (vedi Fig. 1.1), è una superficie interattiva su cui è possibile scrivere, disegnare, allegare immagini, visualizzare testi, riprodurre video od animazioni. I contenuti, visualizzati ed elaborati sulla lavagna, potranno essere quindi digitalizzati grazie ad un software di presentazione appositamente dedicato. La LIM è uno strumento destinato alla didattica d'aula, poiché coniuga la forza della visualizzazione e della presentazione tipiche della lavagna tradizionale con le opportunità del digitale e della multimedialità.

Nell'accezione più comune quando si parla di LIM si intende un dispositivo che comprende una superficie interattiva, un proiettore ed un computer. Oggi l'evoluzione tecnologica offre dispositivi che permettono di sfruttare le

Figura 1.1: Una LIM

potenzialità di uno schermo interattivo e multimediale utilizzando qualsiasi tipo di superficie e pennarello, oppure attraverso schermi "touch screen", anche della grandezza di un normale desktop, che non necessitano di PC e proiettore.

Caratteristiche

La lavagna interattiva multimediale è composta, in prima battuta, dalla superficie interattiva, un dispositivo elettronico avente le dimensioni di una tradizionale lavagna didattica, sul quale è possibile interagire usando le mani o degli appositi pennarelli. Tali superfici si distinguono in tre categorie per quanto concerne la tecnica di visualizzazione.

- Retroproiettate: sono dei grandi schermi collegati ad un personal computer, dotato di software necessario a realizzare i file di presentazione.
- A proiezione frontale: sono collegate con un PC ed un proiettore, che visualizza ciò che viene mostrato sul monitor del computer, su una superficie arbitraria.
- Schermi interattivi: sono di varie dimensioni, tra cui il più grande è il banco interattivo; sono dei display "touch screen" con computer incorporato.

Tutte le tipologie di LIM in commercio sono dotate di un software proprietario che permette l'interattività della superficie ed offre una serie di strumenti

per la realizzazione di presentazioni multimediali. In alcuni casi il software è anche predisposto per leggere determinati formati (scrittura, calcolo, animazioni). Alcuni software permettono di operare sul file in modalità duplice: "scrivania" o "classe", per permettere al docente di preparare la lezione e di presentarla successivamente compiuta ai suoi studenti.

Insieme al software proprietario per realizzare file di presentazione, esistono software compatibili con le LIM. Tali software permettono di condividere lo schermo della lavagna via Web, di interagire con altri PC presenti in classe, di creare *learning object*, mappe concettuali o mentali, simulazioni di vario tipo. Registratori audio e video integrati nel software di presentazione consentono di registrare suoni e filmati, che vengono lanciati sul PC su cui si sta operando, offrendo la possibilità di "filmare" con commento le lezioni realizzate con la LIM. In rete è possibile rinvenire anche software *open-source* capace di operare su qualsiasi modello di lavagna.

L'accessorio principale della LIM è il pennarello, che permette di scrivere od utilizzare i comandi sullo schermo. Alcuni modelli ne possono fare a meno, perchè interagiscono anche con il contatto della mano o sfruttano i pennarelli normali. Il vantaggio del pennarello è la precisione del tocco e del disegno. In alcuni modelli è possibile anche usare i normali pennarelli colorati e cancellabili, perchè la lavagna ed il software riconoscono quanto scritto su qualsiasi superficie. Altri modelli offrono dei telecomandi utili per la risposta (a distanza sulla LIM di tutta la classe) a test costruiti con appositi software.

È possibile anche utilizzare dei mini-schermi da tenere in mano o sulla cattedra. Quanto scritto sulla lavagnetta comparirà sullo schermo grande, per scrivere alla lavagna senza dare le spalle alla classe.

Un'ultima considerazione riguarda le connessioni. La connessione alla rete è assicurata dal PC; alcuni modelli di lavagne sono dotati di autonoma connessione *wireless* e di *bluetooth* per l'interazione con il Web, con altre lavagne a distanza, con altri dispositivi presenti in classe a breve raggio.

Vantaggi

Numerosi studi del mondo anglosassone hanno evidenziato le principali potenzialità dello strumento LIM. I vantaggi riguardano soprattutto: la visualizzazione in grande, l'utilizzo delle tecnologie a favore di tutta la classe, la semplificazione dei concetti, l'interattività, l'aggregazione di risorse multimediali. C'è da considerare, inoltre, il favore degli studenti, l'estrema semplicità d'impiego e la costruzione collaborativa del percorso di studio.

La visualizzazione in grande è la più riconosciuta tra le potenzialità della LIM. Essa permette di presentare una molteplicità di contenuti, utilizzando non più solo l'ascolto o la lettura individuale, ma anche la forza comunicativa dell'immagine.

L'utilizzo delle tecnologie a favore di tutta la classe è una delle opportunità più innovative, perchè permette di emanciparsi dall'impiego individuale delle tecnologie verso una dimensione più relazionale, data dall'essere uno strumento a disposizione di tutta la classe. L'interattività è data da mol-

teplici livelli; riguarda sia la possibilità di intervenire, personalizzandoli, su tutti i file presenti sullo schermo, sia la possibilità anche fisica di agire sulla lavagna, sia, infine, in presenza di collegamento al Web, la possibilità di accedere dalla classe alle risorse di Internet.

L'aggregazione di molteplici risorse multimediali permette di costruire percorsi di insegnamento/apprendimento che sfruttano i nuovi media in ogni dimensione, immagine, audio, video.

Gli studenti "avvertono" la LIM come vicina al loro modo di comunicare e di accedere alle informazioni. L'estrema semplicità di utilizzo è all'origine della diffusione delle LIM. Le competenze necessarie per il suo impiego sono quelle di base (scrittura, apertura ed inserimento file, upload, download, uso del Web).

La costruzione collaborativa dei percorsi di studio fanno della LIM uno strumento particolarmente efficace per la realizzazione di attività di gruppo in classe, non ultime le potenzialità dimostrate dalla LIM nel campo dell'integrazione di alunni stranieri.

La sua dimensione fisica, l'impiego di linguaggi altri dalla semplice scrittura, il riconoscimento dei caratteri e la visione in grande rendono più agevoli e più integrati gli apprendimenti di studenti con diverse abilità o stranieri. L'impiego del Web per la condivisione dei materiali e la comunicazione a distanza, inoltre, rende la LIM utilizzata anche nei contesti di scuola in ospedale.

Confronto

Nel mondo delle tecnologie per la didattica, in questi anni si sono affermati molteplici strumenti, a partire dall'impiego del Web, ma anche dei software di presentazione, dei *learning object*, dei video, delle foto digitali, del *podcasting*. Consideriamo un breve confronto di alcune delle citate tecnologie didattiche con le LIM.

Rispetto al Web, la LIM permette di sfruttare tutte le potenzialità informative del Web stesso e di riutilizzarne i percorsi originali, sia grazie alla possibilità di catturare schermate di siti, che di aggregare link utili per la didattica, creando, così, percorsi di navigazione sulla LIM, che possono funzionare anche in assenza di collegamento alla rete.

Rispetto alla presentazione didattica più o meno animata, quella realizzata con la LIM si presenta come un lavoro in corso. Non ha un carattere autoesplicativo, ma permette di costruire un percorso didattico coinvolgendo gli studenti. Richiede, pertanto, una partecipazione attiva ed è sempre aperta a modifiche successive.

Analogamente, i *learning object* sono dei percorsi di apprendimento chiusi, definiti da una sezione di spiegazione ed una di verifica. La LIM può assemblare entrambe le sezioni, lasciando spazio a possibili modifiche ed integrazioni anche in corso d'opera. Anche i video didattici sono dei percorsi di apprendimento chiusi, non passibili di modifiche se non in fase di edizione e montaggio.

La LIM offre la possibilità di assemblare, nello stesso percorso didattico,

video, link a video, catture tratte da video, nonché di registrare tutto quel che avviene sullo schermo del display e della lavagna in formato video e di riutilizzarlo, quindi, come guida.

Analogo adattamento può essere destinato a qualsiasi immagine o foto digitale, che sulla LIM possono essere corredate di testo sovrapposto, ritagliate e ritoccate. Grazie alla potenzialità di inserire file audio, anche senza possedere un collegamento alla rete, sul software di presentazione della LIM può essere realizzato un *podcast* che assembli file audio realizzati con il microfono del PC o con i lettori MP3.

La LIM ha la possibilità di essere collegata alla rete (attraverso connessioni con fili o senza fili) o di essere utilizzata senza Internet. Il collegamento ad Internet permette, in prima battuta, la realizzazione di percorsi ipermediali. Attraverso software appositi è, inoltre, possibile condividere a distanza il file di presentazione su cui si sta operando. Grazie a questa opportunità ed al collegamento *wireless* o *bluetooth* di cui sono dotate molte lavagne, è possibile far dialogare i PC presenti in classe con la LIM, quest'ultima con una LIM presente in un'altra classe e, a sua volta, con i PC dell'altra classe ad essa collegati.

Nonostante le potenzialità della LIM siano moltiplicate esponenzialmente con l'uso delle connessioni e della rete, anche l'impiego offline può offrire molte chance. La natura di assemblaggio di risorse del software di presentazione della LIM permette una navigazione tra le risorse allegate al file o presenti sul PC su cui si opera, riproducendo, quasi, le funzionalità del Web.

1.4 La didattica della geometria e le tecnologie

Vi è una lunga tradizione dei matematici nel fare uso di strumenti tecnologici; viceversa, l'uso di questi strumenti ha dato luogo a molti problemi matematiciassai interessanti (per esempio riga e compasso per le costruzioni geometriche, logaritmi e strumenti meccanici per il calcolo numerico).

Negli ultimi anni, la nuova tecnologia e, in particolare, i computer, ha riguardato ampiamente tutti gli aspetti della nostra società. Molte attività tradizionali sono diventate superate, mentre nuove professioni e nuove sfide sorgono. Per esempio, il disegno tecnico non è più fatto a mano. Oggi, invece, si usa software commerciale, i plotter ed altri dispositivi tecnologici. Il CAD/CAM ed il software di algebra simbolica cominciano ad essere largamente disponibili.

I computer hanno anche reso possibile costruire la "realtà virtuale" e generare in modo interattivo animazioni o figure straordinarie (ad esempio le immagini frattali). Ancora di più, i dispositivi elettronici possono essere utilizzati per eseguire esperienze che nella vita di tutti i giorni sarebbero inaccessibili, od accessibili solo come risultato di lunghi e spesso noiosi lavori.

In tutte queste attività la Geometria è profondamente coinvolta per accrescere sia la capacità di utilizzare appropriatamente gli strumenti tecnologici, sia quella di interpretare e capire il significato delle immagini prodotte.

I computer possono essere anche usati per guadagnare una più profonda comprensione delle strutture geometriche grazie a software specifici preparati

per scopi didattici. Gli esempi includono la possibilità di simulare le tradizionali costruzioni con riga e compasso, o la possibilità di muovere elementi di base di una configurazione sullo schermo, conservando le relazioni geometriche fisse. Questo può condurre ad una presentazione dinamica degli oggetti geometrici e può favorire l'identificazione delle loro invarianti.

Fino ad ora, la pratica scolastica è stata solo marginalmente influenzata da queste innovazioni. Ma in un prossimo futuro è probabile che almeno qualcuno di questi nuovi argomenti troverà la propria collocazione nei curricoli. Ciò comporterà grandi sfide. Di seguito ne vengono elencate alcune [VILL].

- Come influirà l'uso dei computer nell'insegnamento, nei suoi obiettivi, nei suoi contenuti e nei suoi metodi?

- Saranno conservati, in tal modo, i valori culturali della geometria classica, o si evolveranno? E, se ciò accadrà, come?

- Nei paesi in cui le costrizioni finanziarie non consentiranno una massiccia introduzione di computer nelle scuole in un prossimo futuro, sarà comunque possibile ristrutturare i curricoli, per esempio di geometria, per far fronte alle principali sfide originate da questi dispositivi tecnologici?

Al momento, gli strumenti tecnologici di ausilio alla didattica della geometria sono quasi esclusivamente rappresentati da ambienti di costruzione dinamica libera, come il software proprietario Cabrì Géometre [CABRI] ed il software

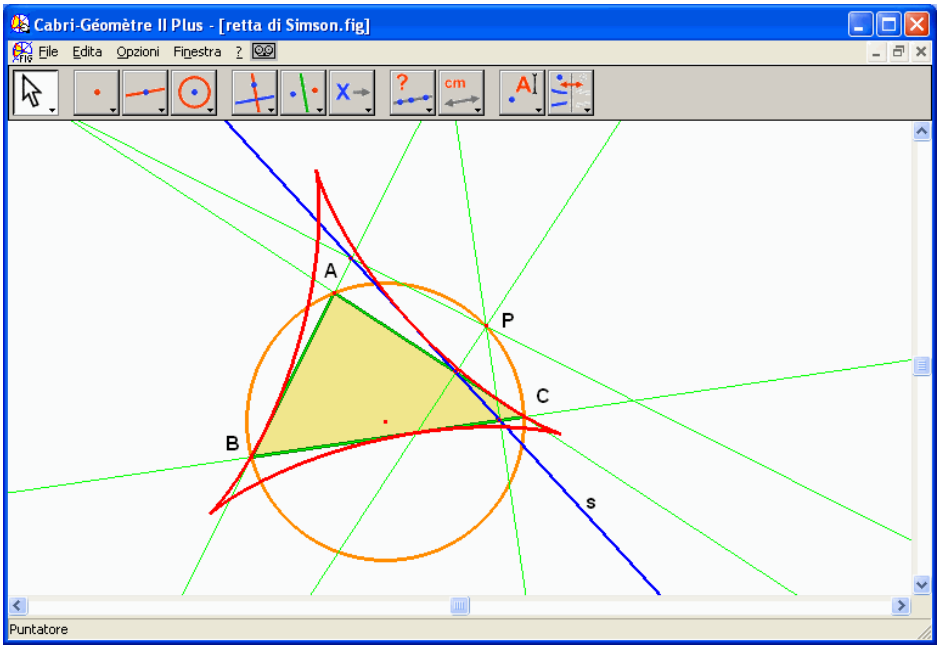


Figura 1.2: Cabri Géomètre II

open source GeoGebra [GEOG], distribuito con licenza GNU General Public License.

Cabri Géomètre II (vedi Fig. 1.2) è un ambiente di grande valore formativo ormai diffuso in tutto il mondo. L'interattività è la caratteristica fondamentale di questo programma. Scopo dell'ambiente è quello di realizzare costruzioni geometriche piane, di qualsiasi complessità, a partire da alcuni oggetti base (come punti, rette, circonferenze e così via) e da alcune costruzioni base (come punto medio, retta perpendicolare, retta parallela e così via). Le costruzioni ottenute sono di tipo dinamico: si può agire su di

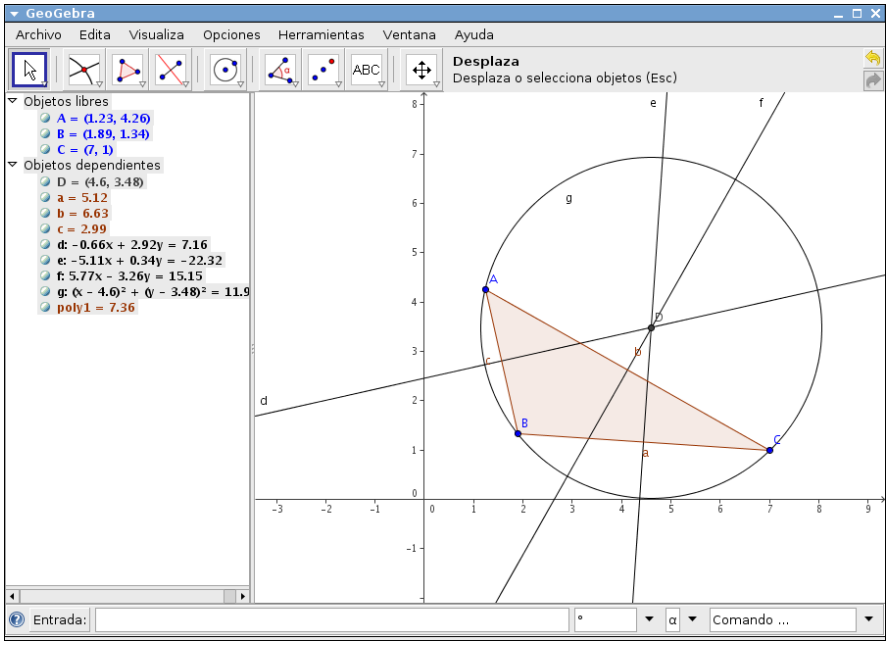


Figura 1.3: GeoGebra

esse trascinando col mouse gli oggetti liberi. L'aspetto essenziale è che le trasformazioni di una figura conservano le relazioni che si sono impostate (ad esempio rette parallele rimangono parallele, i punti medi rimangono tali, e così via) [CABRI].

Per quanto riguarda GeoGebra (Fig. 1.3), questa applicazione, gratuita e multipiattaforma, permette di interagire, in maniera dinamica, con la geometria, l'algebra ed il calcolo numerico. Come programma educativo, GeoGebra ha ricevuto vari premi internazionali e può essere utilizzato nelle scuole primaria e media per rinforzare il processo di apprendimento degli

alunni. Con GeoGebra è possibile realizzare costruzioni geometriche di diversa complessità, a partire da punti, vettori, segmenti e rette, tramite funzioni dinamicamente configurabili [GEOG].

Il software sviluppato in questo lavoro di tesi si intende inserire in un ambito diverso da quello di Cabri e di GeoGebra. Vuole rappresentare un ausilio alla didattica di un argomento preciso e mirato; non uno strumento per "creare", ma uno strumento per "capire". In maniera divertente ed interattiva, esso accompagna il bambino nella spiegazione del calcolo dell'area dei poligoni. Più in dettaglio, mostra il *motivo* delle formule per il calcolo delle aree, aiutando l'alunno non solo a memorizzarle ed utilizzarle nel modo corretto, ma anche a capirne il significato geometrico, aspetto fondamentale per la piena comprensione del percorso didattico sulla geometria piana.

Capitolo 2

Gli strumenti utilizzati

Per lo sviluppo dell'applicazione, oggetto di questa tesi, sono stati utilizzati strumenti diversi. Il progetto è stato scritto in Java ed in Mathematica e sono stati sfruttati MathLink e J/Link, due componenti di Mathematica per la comunicazione con altri linguaggi di programmazione.

2.1 Mathematica

Mathematica è un ambiente di calcolo simbolico e numerico multipiattaforma ed un potente linguaggio di programmazione interpretato. Questo linguaggio è basato sulla riscrittura di espressioni (term-rewriting) e supporta svariati paradigmi di programmazione, tra cui la programmazione funzionale, la programmazione logica, la programmazione basata sul riconoscimento di schemi (pattern-matching) e sulle regole di sostituzione (rule-based), nonché la più

tradizionale programmazione procedurale.

2.1.1 Il linguaggio

In Mathematica, il linguaggio di base viene interpretato da un kernel che esegue l'elaborazione vera e propria; i risultati vengono quindi comunicati ad uno specifico front-end tra quelli disponibili. La comunicazione tra il kernel e questi ultimi (o qualsiasi altro client, ad esempio programmi scritti dall'utente) utilizza il protocollo MathLink, di cui parleremo nella sezione 2.2.1. È possibile che vari processi front-end si connettano allo stesso kernel o che uno stesso front-end sia connesso a kernel differenti.

Di norma, il front-end è rappresentato da un documento di testo interattivo, il notebook (blocco per gli appunti), che è in grado di visualizzare ed interpretare la notazione matematica bidimensionale in formato WYSIWYG[1] ed incorpora i risultati dell'elaborazione sotto forma di testo, formule, grafici, suoni ed animazioni.

2.1.2 Rappresentazione dei dati

Sostanzialmente, la struttura dati fondamentale di Mathematica è l'*espressione*. Ogni singolo elemento del linguaggio, dai tipi di dato alla struttura stessa dei notebook, è un'espressione costituita da un'intestazione (`Head`) e da una

[1]WYSIWYG: What You See Is What You Get ("quello che vedi è quello che ottieni" o "ottieni quanto vedi"). Il significato si riferisce al problema generale di ottenere sulla carta testo e/o immagini che abbiano una disposizione grafica uguale a quella visualizzata sullo schermo del computer.

sequenza di argomenti separati da virgole e racchiusi tra una coppia di parentesi quadre. Il vantaggio di questa rappresentazione è che tutto il contenuto di notebook e package (che a loro volta non sono altro che espressioni) può essere rappresentato con semplici file di testo in formato ASCII, condivisibili tra le diverse piattaforme su cui Mathematica è implementato.

2.1.3 Efficienza

Le routine predefinite (built-in) di Mathematica sono scritte per la maggior parte in linguaggio C e come tali sono particolarmente efficienti dal punto di vista dei tempi di esecuzione. Tuttavia, data l'assenza di tipizzazione in Mathematica, ogni procedura invocata in un pacchetto od in un notebook fa capo a più routine specializzate in linguaggio C, ognuna delle quali gestisce l'esecuzione sulla base dei tipi di dato passati dall'utente.

Per ogni procedura il kernel deve prima verificare che i dati passati siano validi e successivamente scegliere, tra gli algoritmi per il particolare tipo di dati identificato, quello che meglio si adatta a risolvere il problema. Tutto questo comporta un overhead che è in genere trascurabile per la singola procedura, ma diventa rilevante quando siano interessate più procedure in successione o si debbano ripetere i calcoli all'interno di un ciclo.

2.1.4 Mathematica Player

Wolfram Research rende gratuitamente disponibile un software che permette di visualizzare i notebook con la formattazione matematica bidimensionale. Con la versione 6 ne è stata introdotta una versione avanzata denominata *Mathematica Player*, che è in grado di eseguire, in tempo reale, il codice creato da Mathematica e memorizzato in file con estensione .nbp (Notebook Player files). Si tratta di dimostrazioni interattive nelle quali l'utente è in grado di modificare determinati parametri, per mezzo di un'interfaccia grafica realizzata con appositi comandi del linguaggio Mathematica.

Più che un semplice Reader, Mathematica Player contiene per intero il motore di Mathematica, in modo da permettere l'esecuzione di qualsiasi applicazione controllabile tramite interfaccia grafica [MATH].

2.1.5 Pro e contro

Il principale vantaggio dell'utilizzo di Mathematica è la potenza e flessibilità del suo linguaggio di programmazione che, combinando diversi paradigmi, permette di scrivere la stessa applicazione in una molteplicità di approcci diversi, per cui è possibile creare applicazioni molto complesse con poche righe di codice.

Non meno importante risulta anche la sua estrema efficienza, di cui abbiamo parlato precedentemente nella sezione 2.1.3, e la generalizzazione delle funzioni a disposizione del programmatore come built-in di sistema.

Mathematica offre, infatti, funzioni altamente generalizzate per ambito di applicabilità e numero di dimensioni dei dati, delegando ad algoritmi interni la scelta delle procedure ottimizzate per il particolare tipo di dato passato. L'operazione di convoluzione, ad esempio, può essere applicata a liste monodimensionali o multidimensionali di valori numerici o variabili simboliche, mentre la procedura Fourier per il calcolo della FFT (Fast Fourier Transform) presenta la medesima sintassi per i casi mono- e multi-dimensionale. Viene in questo modo semplificata la modifica del codice a problemi più generali. Anche il numero di funzioni disponibili è elevato e permette di scrivere programmi compatti ed al contempo ottimizzati, grazie ad algoritmi interni spesso allo stato dell'arte.

La principale critica che si può rivolgere a Mathematica è che, a differenza di altri ambienti di calcolo simbolico, che sono open source e possono venire liberamente utilizzati, copiati e, all'occorrenza, modificati, Mathematica è un software proprietario che prevede limitazioni riguardo all'uso, la copia e la modifica e che presenta un costo per ogni licenza d'uso talmente alto (nell'ordine dei tremila euro) che lo pone fuori dalla portata della maggior parte dei potenziali utilizzatori.

Un altro difetto è rappresentato dal fatto che i dati manipolati da Mathematica sono principalmente liste annidate, la cui esatta struttura è di difficile discernimento quando il numero di dimensioni cresce.

La natura prevalentemente funzionale della programmazione in Mathematica, infine, rende difficoltosa la ricerca di errori nel codice (debugging):

i messaggi di errore non puntano direttamente all'istanza della funzione che li ha causati e non vengono forniti numeri di riga od altre informazioni che permettano di risalire al punto del codice in cui l'errore si è verificato.

2.2 J/Link e MathLink

2.2.1 MathLink

In molti casi risulta conveniente utilizzare Mathematica in comunicazione ad alto livello con programmi esterni, sfruttando anche l'eventuale scambio di dati. La comunicazione con altre applicazioni avviene attraverso il protocollo MathLink. Esso permette non solo la comunicazione tra il kernel ed i front-end, ma fornisce anche un'interfaccia generale tra il kernel ed una applicazione arbitraria.

Wolfram Research distribuisce gratuitamente un kit di sviluppo per collegare applicazioni scritte in C al kernel di Mathematica attraverso MathLink. Altri due componenti di Mathematica, attraverso MathLink, permettono agli sviluppatori di collegare il kernel ad un programma in Java od un programma .NET. Questi sono J/Link e .NET/Link.

2.2.2 J/Link

J/Link è una libreria di classi che permette di interfacciare codice Java e programmi in linguaggio Mathematica. Da un lato permette ai programmi Java

di utilizzare i comandi di Mathematica per eseguire calcoli; dall'altro viene concesso a Mathematica di caricare classi Java, manipolare oggetti Java ed eseguire chiamate a metodi, rendendo possibile, ad esempio, la costruzione di un'interfaccia grafica per un'esecuzione interattiva del codice Mathematica.

In maniera un po' semplicistica, J/Link può essere considerato come un kit di sviluppo MathLink per Java; una delle sue migliori caratteristiche, infatti, è che, per la maggior parte degli usi, J/Link nasconde completamente il sottostante MathLink, risparmiando al programmatore la necessità di conoscere il funzionamento di quest'ultimo.

Per tutti i tipi di programmi MathLink, J/Link fornisce un ulteriore livello di astrazione rispetto alla tradizionale interfaccia di programmazione MathLink, rendendo di fatto Java il linguaggio di programmazione più semplice e conveniente per la scrittura di applicazioni che interagiscono con Mathematica [JLINK].

2.2.3 Interazione di J/Link e Mathematica Player

Lo sviluppo di applicazioni Java che interagiscono con il motore di Mathematica è reso ancora più interessante dalla possibilità di sfruttare le potenzialità di Mathematica Player, che presenta il vantaggio di essere gratuito, al posto di Mathematica. In questo modo è possibile ottenere un'applicazione estremamente portabile (Mathematica e Mathematica Player sono distribuiti per Windows, MacOS e Linux), liberamente distribuibile e gratuita.

Mathematica Player presenta, ovviamente, delle limitazioni in confronto alla versione completa del software. Non è permessa la creazione di documenti, ma solo l'esecuzione di alcuni di essi. I documenti visualizzabili nel Player devono, infatti, subire un processo di trasformazione, in cui vengono convertiti nel formato apposito, con estensione .nbp. I documenti che possono essere sottoposti a questo processo di conversione devono rispettare alcune linee guida, tra cui quelle elencate di seguito.

- Devono essere evitate tutte le operazioni di MathLink atte a gestire codice di applicazioni esterne; non si possono, quindi, creare oggetti Java nel documento .nbp, né utilizzare kernel remoti.

- Non è possibile utilizzare front-end diversi dall'interfaccia messa a disposizione da Mathematica; non si possono, quindi, scrivere interfacce grafiche in altri linguaggi di programmazione.

- Tutto il contenuto interattivo del notebook deve essere generato da comandi `Manipulate` e, al suo interno, è permesso l'uso di soli elementi *mouse-driven*, quali `Slider`, `Checkbox`, `PopupMenu` e così via.

- Non sono supportate le finestre di dialogo, quali `Input` ed `InputString`.

- La funzionalità di importazione ed esportazione di dati non è disponibile.

Nello svolgimento della presente tesi sono stati effettuati svariati tentativi, per utilizzare Mathematica Player in comunicazione con un'applicazione

Java. Il probema, che è risultato determinante nella decisione di scartare questa ipotesi, è che non è possibile mostrare immagini ottenute con comandi di Mathematica, all'interno di un componente Java. Questa impossibilità deriva direttamente dall'ultimo punto dell'elenco appena citato: la funzione di visualizzazione di immagini, infatti, è ottenuta esclusivamente tramite un'operazione di esportazione, operazione che Mathematica Player non è abilitato ad eseguire.

Per questo motivo il progetto descritto in questa tesi è stato svolto mettendo in comunicazione un'applicazione Java con il programma Mathematica completo, a discapito delle desiderate caratteristiche di libera fruizione [PLAY].

Capitolo 3

Descrizione del progetto

3.1 Dettagli d'integrazione di Java e Mathematica

Le due interfacce più importanti, che il programmatore deve conoscere per integrare Java e Mathematica, sono `MathLink` e `KernelLink`. Verranno descritte in dettaglio nei paragrafi che seguono.

3.1.1 Connessione

`MathLink` rappresenta il protocollo a basso livello che permette di spedire e ricevere espressioni di Mathematica, è utilizzato dal front-end del notebook per comunicare con il kernel ed è sfruttato anche da numerose applicazioni che comunicano con il kernel di Mathematica.

`KernelLink,` invece, è un'estensione di `MathLink` che presenta, in aggiunta, alcune procedure di alto livello, significative solamente se dall'altro lato del link stabilito si trova un kernel di Mathematica. L'idea, infatti, è quella che l'interfaccia `MathLink` gestisca tutte le operazioni che possono essere eseguite con un link, senza fare alcun tipo di ipotesi su quale tipo di programma sia presente all'altro lato del link, mentre l'interfaccia `KernelLink` aggiunge l'assunzione che si stia comunicando con un kernel di Mathematica.

Per la creazione di link si utilizza la classe `MathLinkFactory`, che contiene il metodo `createKernelLink()`. Si utilizza una classe `Factory`, che crea l'istanza di un link, per fare in modo che i client delle classi J/Link rimangano separati dai dettagli dell'implementazione e quindi viene impedita l'istanziazione diretta dei link. Una tipica chiamata su piattaforma MacOsX a questo metodo è la seguente:

```
KernelLink ml = MathLinkFactory.createKernelLink("-linkmode
launch -linkname '\"/Applications/Mathematica.app/Contents/
MacOS/MathKernel\" -mathlink'");
```

Questo metodo restituirà un oggetto di tipo *link*, altrimenti solleverà un'eccezione di tipo `MathLinkException`, per cui è necessario circondarlo con un blocco `try / catch`.

La semplice creazione del link, però, non garantisce che il link sia connesso e funzionante. Esiste una notevole differenza, infatti, tra la creazione di un

link (che riguarda la preparazione dal lato *applicazione*) e la connessione di un link (che verifica se l'altro lato è pronto e reattivo). Il tentativo di connessione verrà automaticamente eseguito non appena si tenterà una lettura od una scrittura sul link, ma è possibile effettuare la connessione anche in maniera manuale con il metodo `connect()`. Al termine di utilizzo del link è necessario ricordarsi di chiuderlo con il metodo `close()`.

MathLink permette di creare e connettersi ad un link instaurato con un'istanza del kernel già in esecuzione oppure con un'istanza remota del kernel di Mathematica. Per prima cosa bisogna eseguire Mathematica sulla macchina remota. Supponendo di essere in un ambiente Unix, si utilizza il comando:

```
math -mathlink -linkmode listen -linkname 1234
-linkprotocol tcpip
```

Nell'applicazione Java, poi, bisogna creare il link con questo comando:

```
KernelLink ml = MathLinkFactory.createKernelLink("-linkmode
connect -linkprotocol tcpip -linkname 1234@remotemachinename");
```

Il difetto più importante di questa tecnica è la necessità di autenticarsi ed eseguire a mano l'istanza del kernel di Mathematica sulla macchina remota. Questo problema, però, può venire aggirato facendo in modo che ciò venga eseguito direttamente dall'applicazione Java, tramite l'utilizzo di un client `rsh` o `ssh`. Questi programmi sono built-in sulle macchine Unix, mentre per

le macchine Windows Mathematica comprende un client apposito, `winrsh`, di cui si mostra di seguito un esempio di connessione [MLINK]:

```
KernelLink ml = MathLinkFactory.createKernelLink("-linkmode
listen -linkprotocol tcpip -linkname 1234");
Runtime.exec("c:\\program files\\wolfram research\\mathematica
\\6.0\\systemfiles\\frontend\\binaries\\windows\\winrs
h -m -q -h -l YourUsername -'math -mathlink -linkmode
connect -linkprotocol tcpip -linkname 1234@localmachinename'");
```

3.1.2 Utilizzo

Una volta instaurato il link con il kernel di Mathematica, ci sono numerosi metodi che permettono di inviare espressioni, in modo che queste vengano valutate dal kernel.

Evaluate

Questo metodo è relativo all'interfaccia `KernelLink` ed incapsula tutti i passi necessari per inviare a Mathematica un'espressione (sotto forma di stringa di caratteri, `String`, o di espressione generica, `Expr`) ed ottenerne un'altra come risposta. Per eseguire questa operazione, il metodo `evaluate()` avvolge l'espressione di input in un `EvaluatePacket`, che viene passato a Mathematica; il risultato ottenuto viene rispedito indietro sotto forma di `ReturnPacket`.

Per leggere ed utilizzare il risultato della computazione è necessario utilizzare il metodo `waitForAnswer()`, che si occupa di estrarre il contenuto dell'oggetto `ReturnPacket`, ed uno dei metodi di tipo *get* per l'estrazione del risultato vero e proprio. Si noti che esiste un metodo *get* per ogni tipo di dato che può essere restituito: `getInteger()`, `getDouble()` e così via.

Un esempio dell'utilizzo di questo metodo è il seguente:

```
evaluate(“3+4”);
waitForAnswer();
int x = getInteger();
```

È importante notare che tutti questi metodi possono sollevare eccezioni di tipo `MathLinkException`, per cui è bene inserirne la chiamata in un blocco `try / catch`.

EvaluateTo

Questa serie di metodi è un'estensione del metodo `evaluate()`. Eseguono l'immissione della funzione da valutare, aspettano che il risultato sia pronto e lo restituiscono come output. Un'ulteriore estensione è rappresentata dal fatto che non sollevano eccezioni nel caso di fallimento della computazione: semplicemente ritornano come risultato `null`; per comprendere quale errore ha causato il fallimento è necessario chiamare il metodo `getLastError()`. Il modo migliore per comprendere appieno ciò che Mathematica e l'applicazione Java si stanno scambiando è utilizzare un `PacketPrinter`: questa

"stampante" scrive sullo *stream* desiderato la descrizione di ogni `Packet` che passa sul link; ciò risulta molto utile in fase di debugging dell'applicazione. Lo statement che segue mostra un esempio di utilizzo di un `PacketPrinter`:

```
PacketListener stdoutPrinter = new PacketPrinter(System.out);
ml.addPacketListener(stdoutPrinter);
```

Nell'esempio qui sopra, `ml` è l'istanza di `KernelLink`.

I metodi appartenenti a questa categoria sono:

```
evaluateToInputForm
evaluateToOutputForm
evaluateToImage
evaluateToTypeset
```

Il metodo `evaluateToInputForm`, in particolare, restituisce una stringa formattata in `InputForm`; analogamente, `evaluateToOutputForm` fornisce il risultato formattato in `OutputForm`. Quest'ultimo formato è molto più *user-friendly*, comprensibile per l'utente; il formato `InputForm`, tuttavia, è indispensabile nel caso ci sia, in seguito, la necessità di passare nuovamente la stringa al kernel di Mathematica per un'ulteriore valutazione.

Il metodo `evaluateToImage` è utilizzato per ottenere come risultato un'immagine ed è utile nel caso si vogliano inserire nell'applicazione Java dei grafici

prodotti da Mathematica. Questo metodo prende in input diversi parametri, quali larghezza, altezza dell'immagine desiderata e la risoluzione (dots per inch). È importante tenere conto del fatto che non basta che una procedura di Mathematica produca un'immagine; per poterla ottenere con `evaluateToImage` è necessario che questa immagine sia *il valore di ritorno* della procedura. Per questo motivo, si può scrivere `evaluateToImage ("Plot[x,{x,0,1}]")`, ma non `evaluateToImage("Plot[x,{x,0,1}];")` perché il punto e virgola fa in modo che il comando ottenga come risultato `Null`.

Il comando `evaluateToTypeset`, infine, è necessario per mostrare all'utente il risultato della valutazione in formato `StandardForm` o `TraditionalForm`[1].

L'output di default di questi due metodi è un array di byte che contiene i dati GIF dell'immagine restituita; in alcuni casi, però, risulta più utile gestire immagini JPEG, per cui è necessario specificare tale formato con il comando:

```
ml.evaluateToOutputForm("$DefaultImageFormat = \"JPEG\"", 0);
```

[1]Il formato StandardForm genera un output che rappresenta le espressioni Mathematica in maniera unica e non ambigua, utilizzabile anche come formato di input. Il formato TraditionalForm, invece, stampa una rappresentazione il più simile possibile alla notazione matematica; questa rappresentazione può presentare, però, ambiguità.

MathCanvas e MathGraphicsJPanel

La classe `MathCanvas` è una sottoclasse di `Canvas` del package AWT, mentre `MathGraphicsJPanel` è una sottoclasse di `JPanel` del package Swing[2] ed entrambe rappresentano un semplice componente grafico in grado di mostrare immagini ottenute da comandi Mathematica. Queste due classi sono concettualmente identiche, mostrano gli stessi metodi per gestire grafici di Mathematica e sono separate solamente per permetterne un facile utilizzo, sia in presenza di componenti AWT, che in presenza di componenti Swing. I metodi principali relativi a queste due classi sono:

```
public void setMathCommand(String cmd);
public void recompute();
public void repaintNow();
public void setUsesFE(boolean useFE);
public void setImage(Image im);
```

Questi metodi si comportano nello stesso modo per entrambe le classi, quindi per brevità verranno spiegati solo in riferimento alla classe `MathCanvas`.

Il comando `setMathCommand` permette di specificare del codice Mathematica, in modo che questo venga valutato ed il risultato venga mostrato all'interno del componente. Analogamente a quanto detto per il metodo `evaluateTo` del kernel, anche per questo metodo non è sufficiente che il

[2] AWT e Swing sono due package Java che gestiscono l'implementazione di interfacce grafiche.

comando produca un grafico: è necessario che il comando di Mathematica fornito sia un comando che ottiene come risultato un grafico. Per questo motivo, si può scrivere `setMathCommand("Plot[x,{x,0,1}]")`, ma non `setMathCommand("Plot[x,{x,0,1}];")` perché il punto e virgola fa in modo che il comando ottenga come risultato `Null`.

Quando viene invocato il metodo `setMathCommand`, l'esecuzione del comando passato come parametro è immediata; l'immagine risultante viene memorizzata in una cache ed utilizzata ogni volta che il componente viene ridisegnato. A volte può capitare, però, che il codice Mathematica che ha prodotto quell'immagine dipenda da qualche variabile che viene modificata in seguito. In questi casi basta utilizzare il metodo `recompute` per forzare la rivalutazione del codice Mathematica e quindi mostrare l'immagine aggiornata.

Quando è necessario modificare un'immagine che viene mostrata ad intervalli molto brevi, risulta, inoltre, utile il metodo `repaintNow`. Esso, infatti, forza la ri-creazione del grafico da mostrare, senza aspettare che tutti i pixel siano pronti. In questo modo è possibile mostrare, in maniera molto fluida, grafici in movimento, che altrimenti mostrerebbero degli scatti di visualizzazione.

J/Link è in grado di creare immagini in due modi: usando esclusivamente il kernel o sfruttando in aggiunta anche dei servizi tipici del front-end in uso. Il metodo `setUsesFE` indica quale dei due modi utilizzare. Dalla versione 6 di Mathematica, però, tutti gli output di tipo grafico necessitano

obbligatoriamente del front-end, per cui questo metodo non ha effetto.

Il comando `setImage` fornisce un modo alternativo per mostrare un'immagine in un componente `MathCanvas`. Con questa alternativa, bisogna creare un oggetto di tipo `Image`, ad esempio utilizzando un metodo `evaluateTo`, ed in seguito passarlo come parametro al metodo `setImage`. Questa soluzione risulta interessante nel momento in cui vi è la necessità di svolgere delle elaborazioni sull'immagine, prima di mostrarla [JLINK].

3.2 Struttura dell'applicazione

L'applicazione sviluppata in questa tesi è costituita da una classe Java che implementa le interfacce `ActionListener` e `ChangeListener` ed estende la classe `JFrame`. In questo modo è possibile fare in modo che la stessa classe mostri gli elementi in una finestra grafica e ne gestisca gli eventi correlati. La classe, inoltre, contiene l'importazione della libreria com.wolfram.jlink

Per prima cosa viene eseguita la connessione al kernel di Mathematica con i comandi visti nel paragrafo 3.1.1. Questo comando è inserito in un blocco `try / catch`, per poter gestire l'eventuale eccezione che può essere sollevata nel caso avvengano errori nel processo di connessione.

Dopo aver creato l'intera interfaccia grafica tramite componenti Swing, le istruzioni successive, che vengono qui di seguito mostrate, sono dedicate al caricamento dei pacchetti Mathematica per la gestione dei poligoni regolari:

```
kern.evaluateToInputForm("Needs[\"" +
```

```
        KernelLink.PACKAGE_CONTEXT + "\"]", 0);
kern.evaluateToInputForm("<< Polytopes`;", 0);
```

`KernelLink.PACKAGE_CONTEXT` contiene semplicemente la stringa `"JLink`"`, ma è preferibile utilizzare tale costante simbolica piuttosto che scrivere la stringa esplicitamente; questa istruzione è necessaria per permettere a Mathematica di caricare il pacchetto relativo alla comunicazione con Java.

Le seguenti istruzioni sono comandi che verrebbero comunque eseguiti al momento della chiamata ad un qualsiasi metodo per l'ottenimento di un'immagine di Mathematica, come ad esempio `evaluateToImage`. È comunque utile eseguirne la chiamata in maniera esplicita il prima possibile, perché si evita il rischio che la finestra del front-end di Mathematica copra graficamente la finestra Java.

```
kern.evaluateToInputForm("ConnectToFrontEnd[]", 0);
this.toFront();
```

3.2.1 Le funzioni Mathematica

Dopo aver effettuato la connessione al kernel, è necessario immettere e valutare le funzioni di Mathematica che verranno richiamate in seguito, nell'esecuzione del codice Java. In questo progetto, le funzioni da valutare sono quelle relative al disegno delle varie figure geometriche da illustrare.

Ogni funzione prende in input un numero intero, che indica la *fase* in cui si trova la spiegazione della figura. La funzione è quindi suddivisa in varie

fasi, che permettono al disegno di evolversi, fino ad illustrare le motivazioni del calcolo dell'area della figura in esame. Qui di seguito viene mostrato lo scheletro tipico di una di queste funzioni:

```
MostraRomboide[step_]:=
Module[{...},
    Switch[
        step,
        1, Graphics[...],
        2, Graphics[...],
        3, Graphics[...],
        4, Graphics[...],
        5, Graphics[...]
    ]
]
```

L'immissione della definizione di queste funzioni avviene con la seguente istruzione:

```
kern.evaluateToInputForm(comando, 0);
```

Nell'istruzione qui sopra, *comando* è una variabile `String` che, di volta in volta, contiene il codice della definizione della funzione. In questo modo, ogni volta che nel codice Java è necessario mostrare il risultato dell'esecuzione di una di queste funzioni, è sufficiente la seguente istruzione:

```
mathCanv.setMathCommand(“MostraRomboide[n]”);
```

In questo statement si assegna al componente `mathCanv`, che è di tipo `MathGraphicsJPanel`, il comando Mathematica per richiamare la funzione `MostraRomboide`, alla fase n.

In questo progetto, le funzioni di Mathematica sono gestite da componenti Java, in modo da poter sfruttare tutta la libertà che Java mette a disposizione per quanto riguarda l'interfaccia grafica. Ogni funzione è controllabile da:

- un componente `Slider` (barra di scorrimento), che permette di mostrare una certa fase della spiegazione, muovendo manualmente il cursore;
- un pulsante Play che aziona la partenza dell'animazione, data dallo scorrimento delle varie fasi (che avviene, così, in maniera automatica);
- un casella `CheckBox`, che permette di indicare se ripetere o no l'animazione, una volta che questa è giunta al termine;
- due pulsanti che permettono di spostarsi di una fase in avanti o indietro nella spiegazione.

Nella Figura 3.1 viene mostrata la finestra principale dell'applicazione Java: si possono notare tutte le componenti grafiche utilizzate, per gestire le immagini restituite dalle funzioni Mathematica.

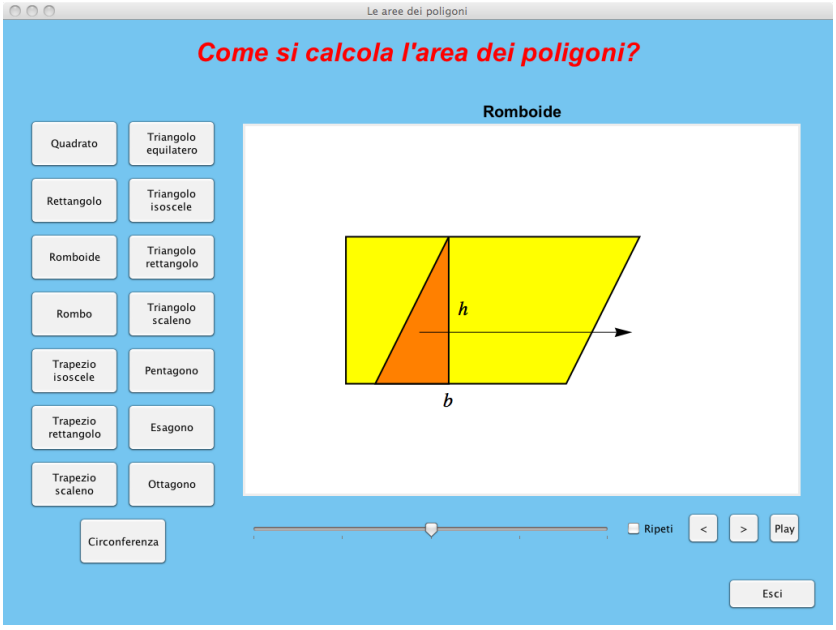


Figura 3.1: La finestra di esecuzione del programma

3.3 Dettagli implementativi per l'usabilità

Un aspetto molto importante di questo progetto è rivestito dalla portabilità dell'applicazione sviluppata. Esistono, infatti, versioni di Mathematica per piattaforme PC, Mac e Unix; Java, dal canto suo, permette una compilazione in quello che viene chiamato `Bytecode`, che può essere eseguito dalla macchina virtuale Java, installabile sul qualsiasi piattaforma.

Date le notevoli differenze tra le piattaforme citate, sono necessari alcuni accorgimenti da attuare nel codice Java dell'applicazione, per fare in modo di utilizzare lo stesso codice in tutti gli ambienti.

Durante la connessione al kernel, è necessario accedere alla directory che contiene l'applicazione del kernel di Mathematica. Dato che l'accesso alle

directory è diverso da piattaforma a piattaforma, bisogna prima verificare su quale sistema operativo si sta lavorando. Questa informazione è ottenuta mediante il metodo Java:

```
System.getProperty("os.name")
```

Con questa informazione è possibile effettuare una connessione diversa per ogni piattaforma considerata. In particolare, il codice dell'applicazione sviluppata, che svolge questo compito, è il seguente:

```
try {

    if (System.getProperty("os.name").contains("Mac"))

        kern = MathLinkFactory.createKernelLink("-linkmode
        launch -linkname '\"/Applications/Mathematica.app/
        Contents/MacOS/MathKernel\" -mathlink'");
    else if (System.getProperty("os.name").contains("Linux"))

        kern = MathLinkFactory.createKernelLink("-linkmode
        launch -linkname 'math -mathlink'");
    else if (System.getProperty("os.name").contains("Windows"))

        kern = MathLinkFactory.createKernelLink("-linkmode
```

```
            launch -linkname 'c:\\program files\\wolfram research\\
            mathematica\\mathkernel.exe'");
        kern.discardAnswer();
    } catch (MathLinkException e) {
        System.out.println("An error occurred `` +
            ``connecting to the kernel.");
        e.printStackTrace();
        if (kern != null) kern.close();
        return;
    }
```

Un aspetto altrettanto importante, per l'usabilità del software, è la possibilità, per l'utente, di eseguire l'applicazione nella maniera più semplice possibile. Ci troviamo, infatti, di fronte ad un utente-tipo rappresentato da un bambino in età scolare, oppure da un'insegnante di scuola primaria, a cui non possono essere richieste conoscenze informatiche particolari per poter utilizzare questa applicazione. È, quindi, assolutamente fondamentale che questo software sia avviabile mediante un semplice doppio click.

Per ottenere questo risultato, di norma sarebbe sufficiente esportare l'applicazione Java in un file di tipo JAR (Java ARchive). In questo progetto, purtroppo, non basta questa semplice operazione. Ci troviamo, infatti, di fronte ad un'applicazione che effettua l'importazione di una libreria esterna,

JLink, che non può essere, a sua volta, inserita in un archivio JAR, in quanto si tratta essa stessa di un archivio di tipo JAR.

Per sopperire a questa mancanza, è stato utilizzato un tool particolare, denominato FatJar [FATJ]. Questo strumento è disponibile sottoforma di plug-in per l'ambiente di sviluppo Eclipse e si occupa specificatamente dell'aggiunta di un file JAR all'interno di un altro file JAR.

Il file JAR che viene creato in questo modo, contiene tutte le classi di tutti i file JAR di riferimento e tutte le classi del progetto. Con questa operazione l'archivio può essere eseguito ovunque, senza dover selezionare alcun *classpath*, poiché tutte le librerie necessarie sono estratte in FatJar.

Dopo lo svolgimento delle operazioni suddette, si ottiene un'applicazione che risulta contenuta totalmente in un unico file, che non necessita di installazione e che può essere eseguito su tutte le piattaforme, mediante un semplice doppio clic. In questo modo, l'utilizzo di questa applicazione risulta estremamente semplice, sia per gli alunni che per gli insegnanti.

Capitolo 4

Conclusioni

In questa tesi è stata presentata un'applicazione di ausilio alla didattica della geometria nella scuola primaria. È stata implementata utilizzando l'integrazione di Java e Mathematica, in modo da combinare la completezza di gestione dell'interfaccia grafica, messa a disposizione da Java, e la potenza di Mathematica nel gestire la creazione delle immagini.

Il suo utilizzo ottimale è quello effettuato in un contesto di gruppo, in cui l'insegnante accompagna le varie fasi dell'applicazione con spiegazioni orali, e con strumenti che permettano di mostrarne l'esecuzione all'intera classe di studenti.

Lo strumento ideale, attraverso cui illustrare l'esecuzione del software alla classe, è rappresentato dalla LIM (Lavagna Interattiva Multimediale), ma la spiegazione effettuata con quest'applicazione risulta efficace anche con un semplice videoproiettore.

L'utilizzo dell'applicazione sviluppata in questa tesi permette ai bambini di comprendere a fondo il calcolo delle aree dei vari poligoni. Attraverso la dinamicità ed interattività della spiegazione proposta in questo progetto, l'alunno è facilitato nella comprensione del motivo del calcolo delle aree; è stimolato nell'apprendimento dalle animazioni, che risultano più consone al modo di pensare del bambino, rispetto alle immagini statiche.

I maggiori pregi presentati da quest'applicazione sono:

- la portabilità;
- la facilità di utilizzo.

La portabilità dell'applicazione qui descritta è determinata dal fatto che il codice Java è eseguibile su qualsiasi macchina ed in presenza di qualsiasi sistema operativo, mentre Mathematica è un'applicazione che può essere installata sulle tre piattaforme principali presenti al momento: Windows, MacOS e Linux.

L'estrema facilità di utilizzo che caratterizza il software sviluppato è stata ottenuta, invece, tramite la possibilità di "compattare" tutte le sue componenti in un unico archivio eseguibile. In questo modo, il programma risulta essere eseguibile con lo stesso metodo su tutte le piattaforme, cioè tramite un semplice doppio clic. Questo ne permette quindi l'utilizzo anche da parte di utenti che non posseggano particolari conoscenze informatiche, quali possono essere alunni ed insegnanti della scuola primaria.

Rimangono, tuttavia, presenti alcuni difetti, che affliggono l'applicazione oggetto di questa tesi: il principale è rappresentato dal fatto che per utilizzare il software è necessario aver installata, sul proprio computer, la versione completa di Mathematica, che è a pagamento. Questo è un problema causato dalle limitazioni imposte dalla Wolfram alle capacità di Mathematica Player, che risulta ottimale per tutte quelle operazioni di calcolo che escludono, però, le operazioni sulla grafica.

Nel caso in cui, in futuro, la società decida di rendere disponibili in Mathematica Player quelle operazioni che permettono la gestione dello scambio di immagini tra il kernel di Mathematica ed un'altra applicazione, il software sviluppato perderebbe all'istante il difetto citato, per diventare un'applicazione totalmente gratuita, oltre che liberamente distribuibile, quale è già adesso.

L'applicazione, presentata in questa tesi, è stata già utilizzata in due scuole primarie di Bologna, riscuotendo un forte successo.

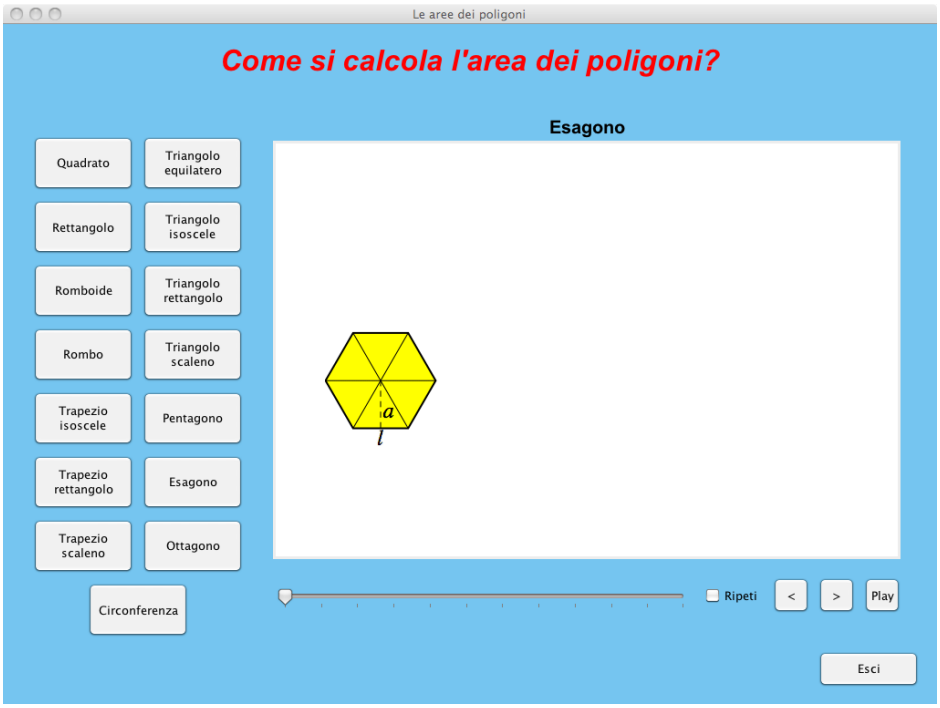
Al momento, è in fase di studio la possibilità di generalizzare il software dalla geometria piana (aree) a quella solida (volumi). Entrambi i software verranno tradotti in inglese e verranno sottoposti al giudizio della Wolfram, per far parte del *Wolfram Demonstration Project* [DEM], che è un archivio di risorse *free* di visualizzazione interattiva, prodotte da ricercatori di varie aree scientifiche e di istituzioni sia pubbliche che private. Nel [DEM] sono raccolti e mantenuti, in modo strutturato, pacchetti liberamente scaricabili, dediti alla soluzione di problemi diversi, dall'educazione nella scuola prima-

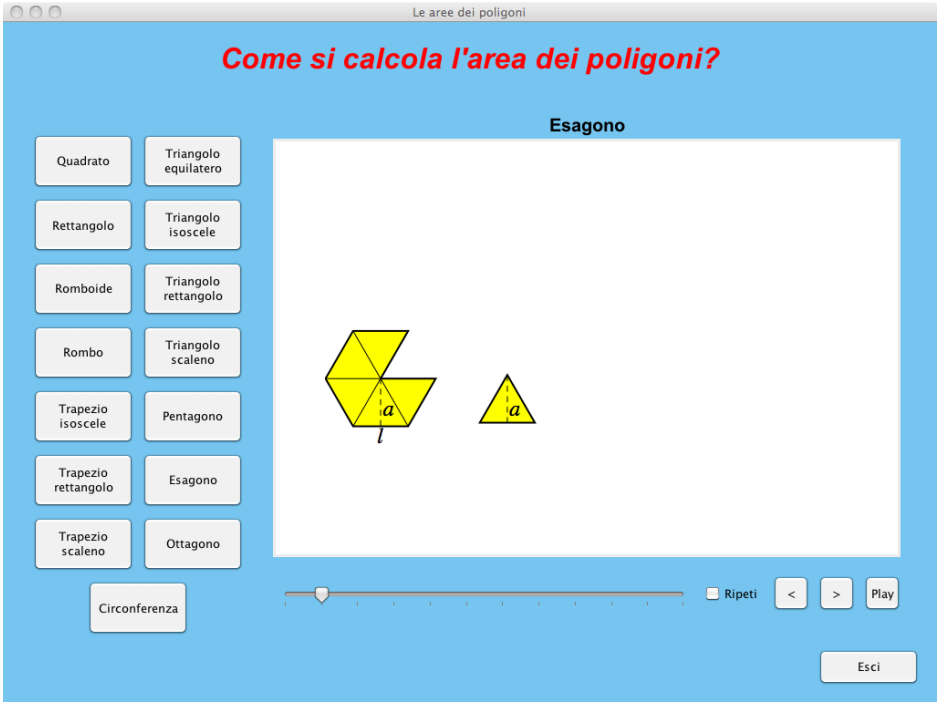
ria, appunto, fino alle frontiere della ricerca. Per il momento, l'applicazione sviluppata in questa tesi e la versione tradotta in inglese verranno, a breve, messe a disposizione per il download gratuito, sul sito www.lannaronca.it.

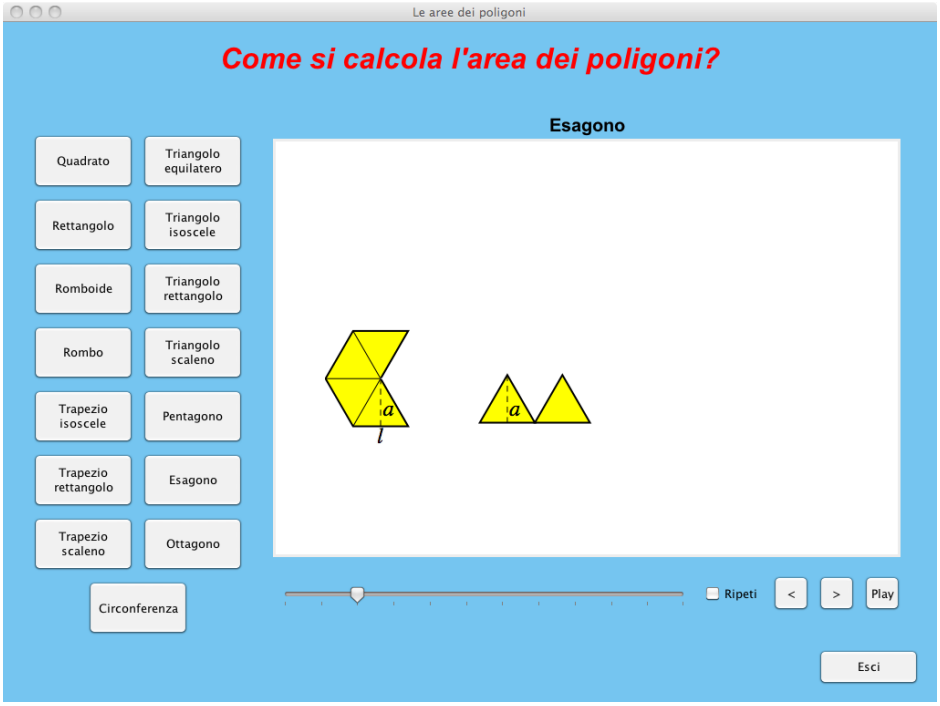
Appendice A

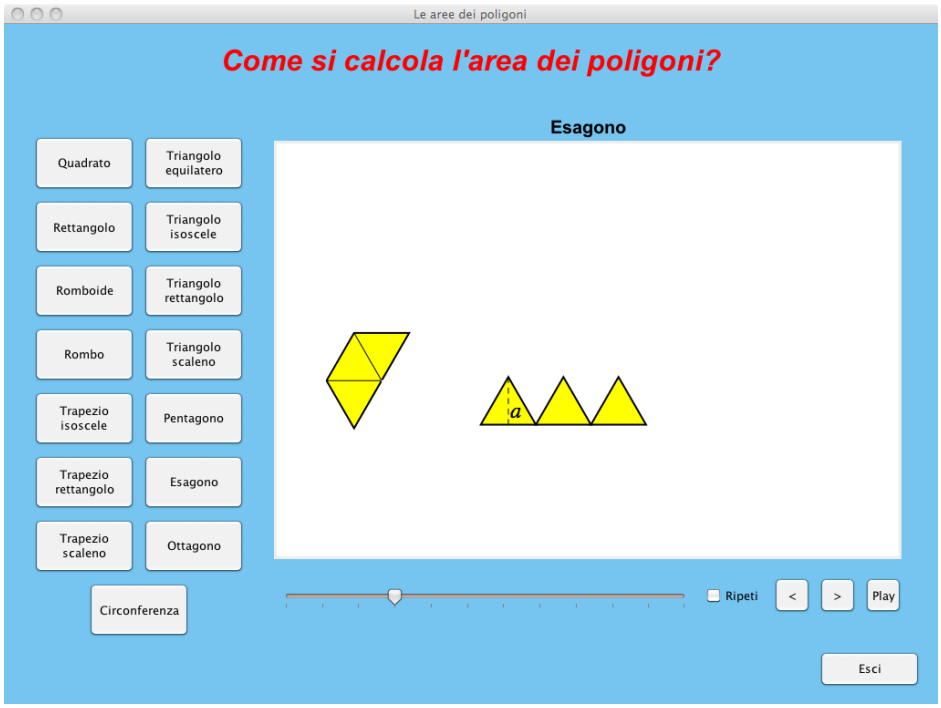
Esecuzione del programma

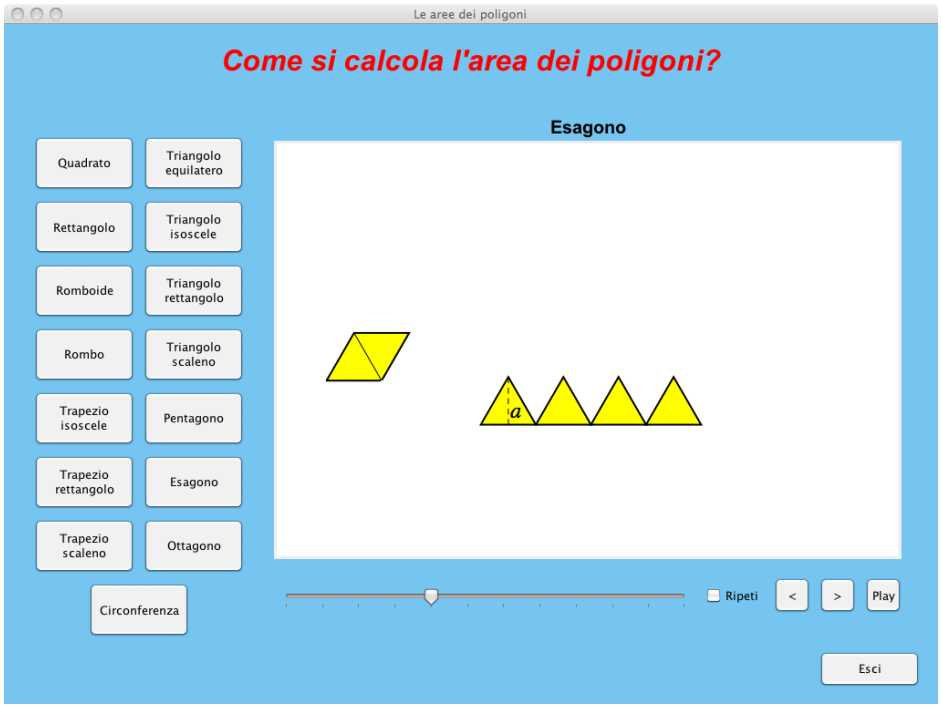
In questa appendice verranno mostrati degli esempi di esecuzione del software, mediante alcuni *screen-shot* della finestra del programma.

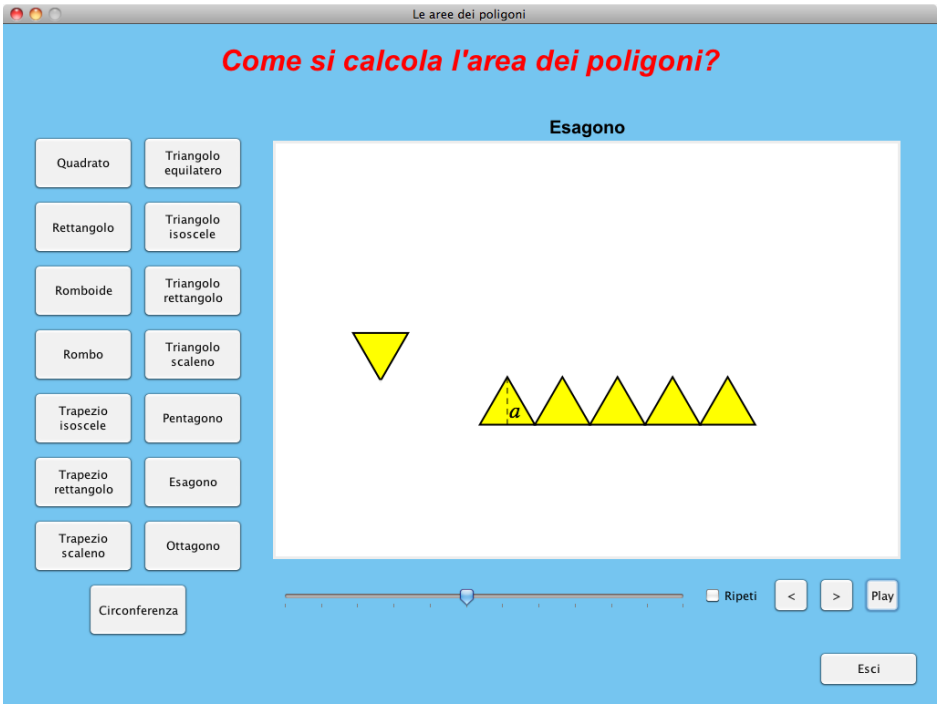


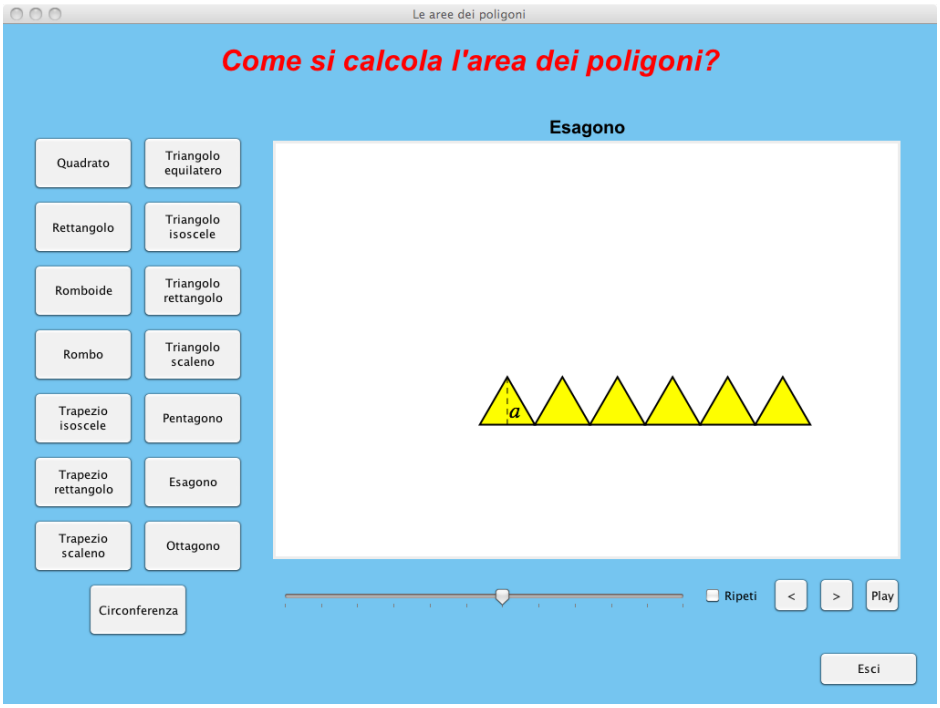

Le aree dei poligoni
Come si calcola l'area dei poligoni?
Esagono
Quadrato
Triangolo equilatero
Rettangolo
Triangolo isoscele
Romboide
Triangolo rettangolo
Rombo
Triangolo scaleno
Trapezio isoscele
Pentagono
Trapezio rettangolo
Esagono
Trapezio scaleno
Ottagono
Circonferenza
a
l
a
Ripeti
<
>
Play
Esci

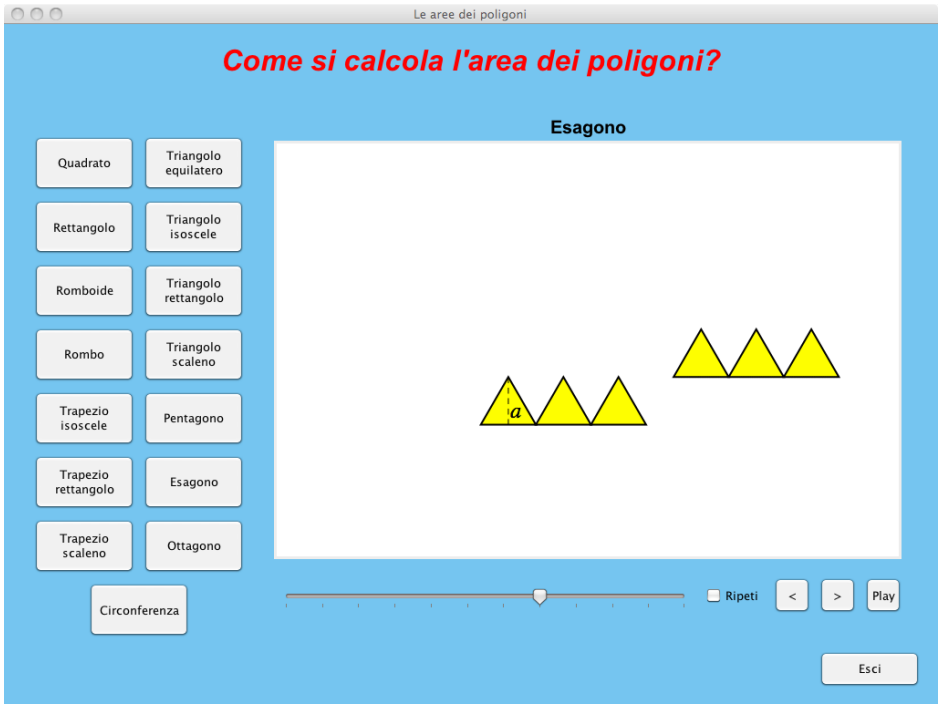

Le aree dei poligoni
Come si calcola l'area dei poligoni?
Esagono
Quadrato
Triangolo equilatero
Rettangolo
Triangolo isoscele
Romboide
Triangolo rettangolo
Rombo
Triangolo scaleno
Trapezio isoscele
Pentagono
Trapezio rettangolo
Esagono
Trapezio scaleno
Ottagono
Circonferenza
a
l
a
Ripeti
<
>
Play
Esci

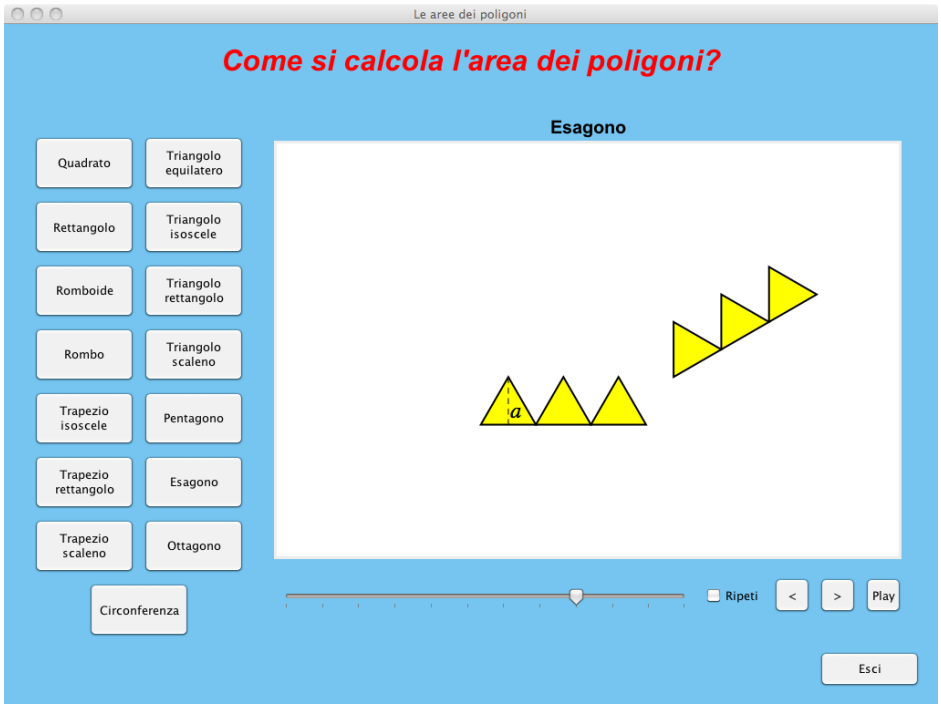

Le aree dei poligoni
Come si calcola l'area dei poligoni?
Esagono
Quadrato
Triangolo equilatero
Rettangolo
Triangolo isoscele
Romboide
Triangolo rettangolo
Rombo
Triangolo scaleno
Trapezio isoscele
Pentagono
Trapezio rettangolo
Esagono
Trapezio scaleno
Ottagono
Circonferenza
a
Ripeti
<
>
Play
Esci

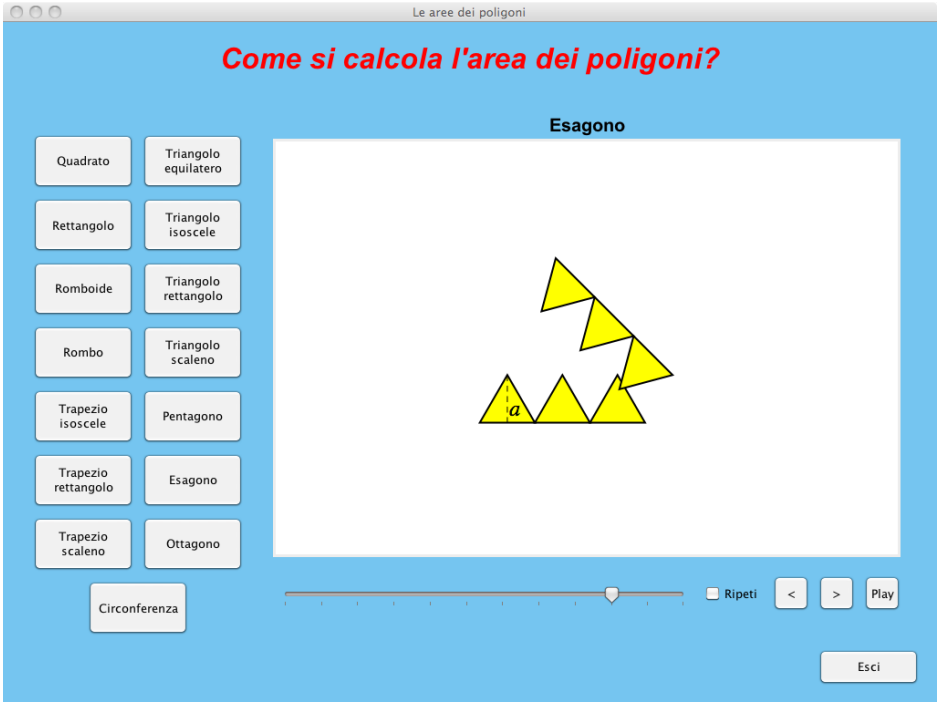

Le aree dei poligoni
Come si calcola l'area dei poligoni?
Esagono
Quadrato
Triangolo equilatero
Rettangolo
Triangolo isoscele
Romboide
Triangolo rettangolo
Rombo
Triangolo scaleno
Trapezio isoscele
Pentagono
Trapezio rettangolo
Esagono
Trapezio scaleno
Ottagono
Circonferenza
a
Ripeti
<
>
Play
Esci


Le aree dei poligoni
Come si calcola l'area dei poligoni?
Esagono
Quadrato
Triangolo equilatero
Rettangolo
Triangolo isoscele
Romboide
Triangolo rettangolo
Rombo
Triangolo scaleno
Trapezio isoscele
Pentagono
Trapezio rettangolo
Esagono
Trapezio scaleno
Ottagono
Circonferenza
a
Ripeti
<
>
Play
Esci

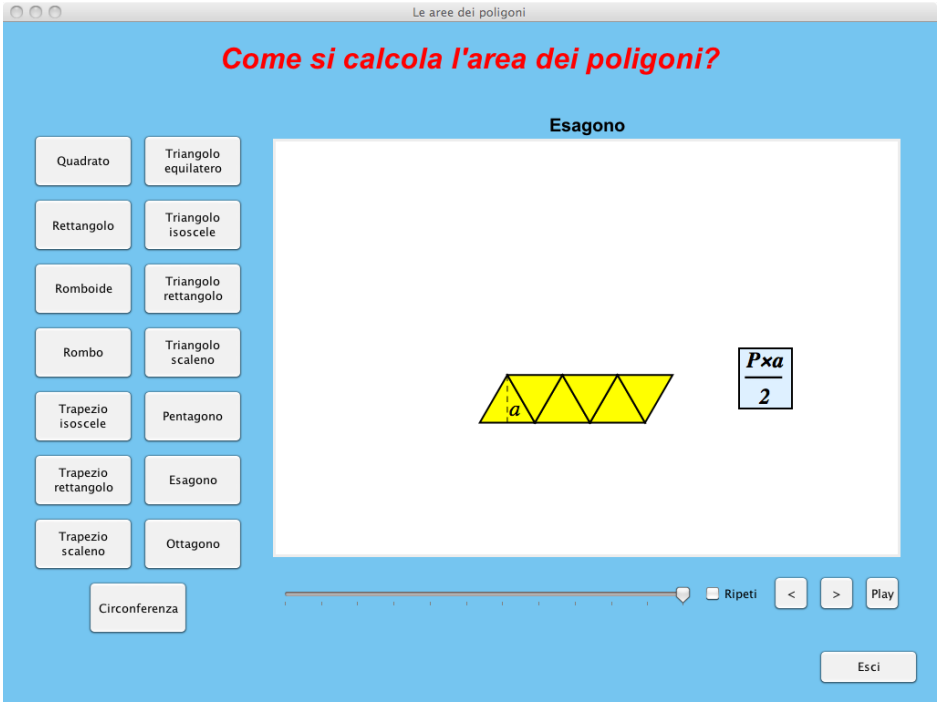

Le aree dei poligoni
Come si calcola l'area dei poligoni?
Esagono
Quadrato
Triangolo equilatero
Rettangolo
Triangolo isoscele
Romboide
Triangolo rettangolo
Rombo
Triangolo scaleno
Trapezio isoscele
Pentagono
Trapezio rettangolo
Esagono
Trapezio scaleno
Ottagono
Circonferenza
a
Ripeti
<
>
Play
Esci


Le aree dei poligoni
Come si calcola l'area dei poligoni?
Esagono
Quadrato
Triangolo equilatero
Rettangolo
Triangolo isoscele
Romboide
Triangolo rettangolo
Rombo
Triangolo scaleno
Trapezio isoscele
Pentagono
Trapezio rettangolo
Esagono
Trapezio scaleno
Ottagono
Circonferenza
a
Ripeti
<
>
Play
Esci


Le aree dei poligoni
Come si calcola l'area dei poligoni?
Esagono
Quadrato
Triangolo equilatero
Rettangolo
Triangolo isoscele
Romboide
Triangolo rettangolo
Rombo
Triangolo scaleno
Trapezio isoscele
Pentagono
Trapezio rettangolo
Esagono
Trapezio scaleno
Ottagono
Circonferenza
a
Ripeti
<
>
Play
Esci


Le aree dei poligoni
Come si calcola l'area dei poligoni?
Esagono
Quadrato
Triangolo equilatero
Rettangolo
Triangolo isoscele
Romboide
Triangolo rettangolo
Rombo
Triangolo scaleno
Trapezio isoscele
Pentagono
Trapezio rettangolo
Esagono
Trapezio scaleno
Ottagono
Circonferenza
a
Ripeti
<
>
Play
Esci


Le aree dei poligoni
Come si calcola l'area dei poligoni?
Esagono
Quadrato
Triangolo equilatero
Rettangolo
Triangolo isoscele
Romboide
Triangolo rettangolo
Rombo
Triangolo scaleno
Trapezio isoscele
Pentagono
Trapezio rettangolo
Esagono
Trapezio scaleno
Ottagono
Circonferenza
a
Ripeti
<
>
Play
Esci



L'esecuzione ha avuto inizio dalla pressione del pulsante Play, che ha causato la partenza dell'animazione, di cui qui sono state mostrate tutte le fasi.

Nell'esempio mostrato si può notare come avvenga la spiegazione grafica del calcolo dell'area dell'esagono. Questa spiegazione riconduce, per passi successivi, l'immagine dell'esagono all'immagine del romboide. Risulta, quindi, utile mostrarla solo dopo aver spiegato agli alunni come avviene il calcolo dell'area di quest'ultima figura.

Bibliografia

[IND] *Indicazioni Nazionali per i Piani di Studio Personalizzati nella Scuola primaria*. Dal Decreto legislativo 19 febbraio 2004, n.59 - (G.U. n.51 del 2-3-2004 - Suppl. Ord. n.31)

[MID] *Argomenti di tecnologie didattiche – Idee, pratiche e strumenti innovativi per l'apprendimento*. Vittorio Midoro, Edizioni Menabò Didattica, 1998

[VILL] *Perspectives on the teaching of geometry for the 21st century: an ICMI study*. Vinicio Vellani. Edizioni Springer, 1998.

[CABRI] http://www.cabri.com

[GEOG] http://www.geogebra.org

[MATH] http://www.wolfram.com/mathematica/

[JLINK] *J/Link User Guide*. Wolfram Research, Inc. 2008

[PLAY] http://www.wolfram.com/solutions/interactivedeployment/ technicalguidelines.html

[MLINK] *A MathLink Tutorial.* Wolfram Research, Inc. http://library.wolfram.com/infocenter/TechNotes/174/

[FATJ] http://sourceforge.net/projects/fjep/

[DEM] http://demonstrations.wolfram.com/