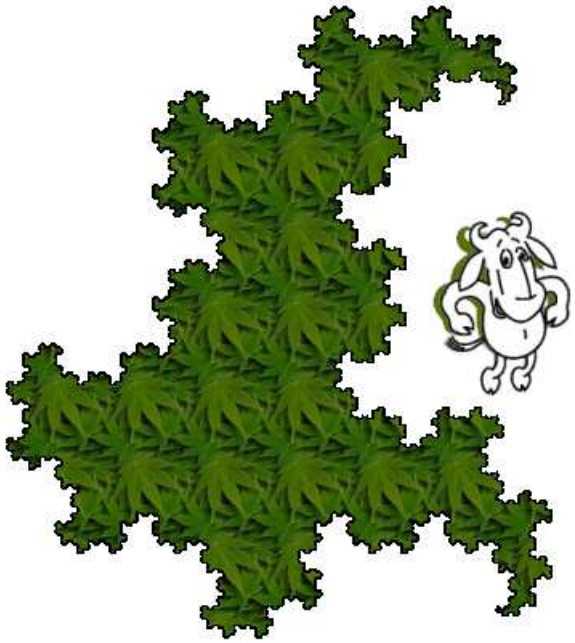
Scheme per Dr. Geo

Andrea Centomo
Hilaire Fernandes



Edizioni Ofset 2005

Introduzione

Con il termine **software libero** si identificano, a grandi linee, tutte le risorse informatiche che vengono rilasciate con codice sorgente aperto, accompagnate da un copyright che ne sancisce la proprietà intellettuale e con licenza GPL (GNU General Public License) o derivata, in modo da renderle liberamente copiabili, utilizzabili e modificabili da parte di tutti, impedendo allo stesso tempo che su di esse vengano aggiunte restrizioni che ne limitino l'uso libero. Alla luce di questa definizione si comprende come il termine **libero** non possa in alcun modo essere assimilato, come spesso accade riduttivamente, a gratuito. Limitarsi al solo aspetto della gratuità, che in alcuni casi riguarda anche altro tipo di software, tralasciando aspetti cruciali come la disponibilità del codice sorgente e la natura della licenza GPL, non permette di cogliere pienamente quel modo originale di pensare l'informatica, tipico dell'approccio libero, che fa della collaborazione e della cooperazione internazionale il perno essenziale del suo stesso sviluppo.
La condivisione della conoscenza, principalmente attraverso l'apertura del codice sorgente, e la collaborazione internazionale attraverso infrastrutture digitali individuano due condizioni imprescindibili per l'esistenza delle risorse libere. Allo stesso tempo tuttavia esse rappresentano obiettivi centrali anche per un'istituzione di carattere culturale ed educativo come la scuola.
Il percorso didattico che proponiamo in questo libro ha come oggetto la **programmazione contestualizzata in un ambiente di geometria dinamica**. L'ambiente in cui si opera, rappresentato dal software libero Dr. Geo, offre una potente integrazione tra motore di manipolazione dinamica degli oggetti geometrici e linguaggio di programmazione di alto livello Scheme. Ambienti in cui geometria e programmazione trovano un punto di incontro non sono una novità nell'esperienza didattica. Si pensi all'esempio del Logo il cui uso, per motivi che ci sfuggono, è rimasto tristemente confinato all'ambito del ciclo primario. Tuttavia il fatto che si possa disporre di un ambiente di programmazione in un ambiente di manipolazione dinamica degli oggetti geometrici rappresenta un aspetto di sicuro interesse pedagogico.
Oltre a questo osserviamo che il lettore, che al termine di questo percorso fosse desideroso di approfondire la conoscenza del linguaggio Scheme e volesse dedicarsi solo alla programmazione, può fruttuosamente utilizzare

l'ambiente Dr. Scheme `http://www.plt-scheme.org` concepito espressamente per la didattica della programmazione Scheme.
Questo libro è stato scritto sulla spinta di diverse motivazioni. La prima è di mostrare che il *software libero* offre strumenti eccellenti per lo studio della geometria e dell'informatica, alternativi a quelli attualmente più diffusi nella scuola. Il secondo, consequenziale al primo, è di incoraggiare le persone all'uso di risorse libere. Ciò non solo per banali ragioni di abbattimento dei costi ma per favorire la condivisione della conoscenza e la crescita di un'informatica collaborativa. Un'ultima importante ragione alla base della realizzazione di questo libro è legata al piacere, che speriamo investa anche il lettore, di esplorare argomenti di matematica affascinanti come, per citare qualche esempio, quello delle tassellazioni, degli insiemi autosimili e della botanica algoritmica.

Capitolo 1

Geometria interattiva

Dr. Geo è un software *libero* progettato per lo studio interattivo della geometria euclidea piana e per un primo approccio ai fondamenti della programmazione in linguaggio Scheme. Nessuno di questi due aspetti, preso singolarmente, rappresenta una novità nel campo del software per l'educazione: gli ambienti per lo studio interattivo della geometria (Kig, Kseg, Cabri, Sketchpad ...), così come i linguaggi di programmazione orientati alla geometria (Logo), hanno già una tradizione abbastanza consolidata nella didattica della scuola italiana. Tuttavia, con l'eccezione dell'ambiente di programmazione Python per Kig, non ci sono noti esempi di software in cui l'integrazione tra ambiente di programmazione e motore geometrico sia così efficace e profonda come in Dr. Geo.

1.1 Funzionalità base

Le funzionalità base di Dr. Geo possono essere presentate nella seguente articolazione:

Oggetti di Dr. Geo

- punto e numero;
- retta, semiretta, segmento, circonferenza, arco, punto (date le coordinate), coordinate di un punto;
- punto medio, vettore, retta parallela, retta perpendicolare, punti di intersezione.

In fase di progettazione del software si è volutamente scelto di non aumentare il numero degli oggetti per stimolare l'utente a fare questo autonomamente. Trattandosi di un software *libero* si hanno due strade per raggiungere questo risultato: utilizzare gli strumenti avanzati messi a disposizione dal programma, che analizzeremo nel seguito, o modificare il codice sorgente. In Dr. Geo sono disponibili altri due oggetti speciali:

- il luogo: per lo studio di luoghi geometrici;
- il poligono: per disegnare poligoni di cui sono assegnati i vertici.

La particolarità di questi oggetti risiede nel fatto che su essi non sono eseguibili alcune operazioni normalmente eseguibili sugli altri oggetti. Ad esempio non è possibile intersecare due poligoni.

Strumenti

- strumenti di misura: righello e goniometro;
- strumenti di trasformazione: isometrie e omotetie;
- strumento di movimento: deformazione dinamica degli oggetti;
- strumenti di visualizzazione: gomma, stile oggetti, zoom;
- strumenti di editing: foglio di testo

Gli strumenti di trasformazione permettono un approccio moderno allo studio della geometria ossia per trasformazioni. Lo strumento di movimento è ciò che caratterizza invece l'aspetto dinamico del software e che permette all'utilizzatore di deformare la figura geometrica mantenendo inalterate le sue caratteristiche geometriche. Gli strumenti di visualizzazione permettono poi di modificare l'aspetto dei diversi oggetti rispetto al comportamento predefinito nella configurazione delle PREFERENZE di Dr. Geo [2]. Dr. Geo mette a disposizione anche un ambiente per editare semplici testi. Il testo viene scritto su un foglio di lavoro, analogo nell'aspetto al foglio su cui si eseguono le costruzioni, e l'utilizzatore può passare da foglio di testo a foglio di costruzione cliccando con il mouse su una linguetta.

1.2 La mia prima figura

La prima figura che costruirai con Dr. Geo è l'asse di un segmento AB; per fare questo ricordati che l'asse di un segmento è la retta perpendicolare al segmento che passa per il suo punto medio. Prima di passare alla costruzione ti ricordiamo che:

Dr. Geo ha un ***manuale in linea*** *che ti può essere utile per molti motivi. Il manuale si attiva cliccando sull'icona che rappresenta un salvagente: è buona norma quando si incontra un problema abituarsi a consultare il manuale per risolverlo autonomamente; solo dopo aver consultato il manuale, se il problema persiste, puoi chiedere aiuto al prossimo. Ricorda anche che il Manuale di Dr. Geo è scaricabile in diversi formati dal sito ufficiale del progetto Dr. Geo* `http://www.ofset.org/drgeo` *e che molti materiali e articoli su questo software sono disponibili in Internet.*

Per costruire l'asse puoi procedere come segue:

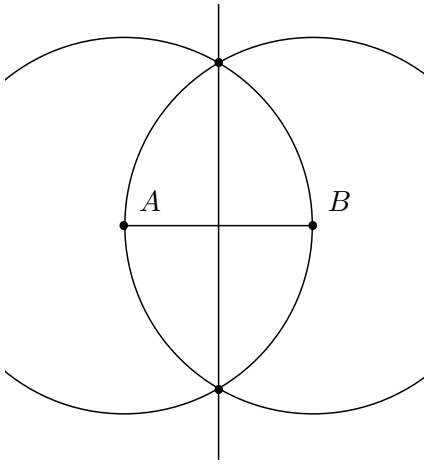


Figura 1.1: Costruzione dell'asse del segmento AB

1. apri Dr. Geo
2. apri una nuova figura FILE → NUOVO → FIGURA
3. disegna due punti sul foglio e assegna ad essi i nomi A e B
4. congiungi i due punti con un segmento
5. disegna la circonferenza di centro A passante per B
6. disegna la circonferenza di centro B passante per A
7. determina con il comando INTERSEZIONE i due punti di intersezione tra le due circonferenze
8. traccia la retta passante per questi due punti.

Se ora provi a deformare dinamicamente la figura 1.1, spostando ad esempio il punto A, noterai che nonostante gli oggetti geometrici cambino la loro forma, le relazioni geometriche che intercorrono tra loro si mantengono invariate. In altri termini noterai che mentre il segmento AB cambia la costruzione restituisce sempre il suo asse.

Esercizio 1 *Costruisci con Dr. Geo baricentro e ortocentro di un triangolo qualsiasi.*

1.3 Macro-costruzioni

Le macro-costruzioni rappresentano uno strumento interno a Dr. Geo utile per ampliare l'insieme dei suoi oggetti di base. Attraverso una macro-

costruzione puoi aggiungere a Dr. Geo una nuova costruzione che per qualche ragione ti interessa. Come in ogni costruzione geometrica si definiscono un certo numero di oggetti geometrici in ingresso e si ottengono alcuni oggetti geometrici in uscita. L'utilizzatore di Dr. Geo stabilisce gli oggetti iniziali e finali della macro-costruzione, che poi verrà salvata in un file con estensione `.mgeo`.
La costruzione di una macro deve essere logicamente coerente: gli oggetti finali della macro devono dipendere in modo esclusivo dagli oggetti iniziali in modo che il software possa ricostruire, senza ambiguità, l'intera costruzione. Da questo punto di vista Dr. Geo è in grado di memorizzare la sequenza della costruzione e di riprodurla ogni volta che l'utilizzatore, dopo aver azionato il comando di esecuzione di una macro, andrà a cliccare su oggetti geometrici che coincidono con gli oggetti in ingresso della macro stessa.

Esercizio 2 *Dopo aver studiato il paragrafo dedicato alle macro, che trovi nel capitolo Funzioni Avanzate del manuale in linea di Dr. Geo [2], scrivi una macro che costruisce un triangolo equilatero di lato dato e salvala nella tua cartella personale Macro con nome* `triangoloequilatero.mgeo`*.*

1.4 Teorema di Aubel

In Matematica si incontrano spesso delle proposizioni che prendono il nome di **teoremi**. Un teorema è una proposizione che è conseguenza logica di altre proposizioni precedentemente dimostrate o assunte come vere (postulati). All'interno di un teorema si riconoscono **ipotesi** e **tesi**. Le ipotesi sono le assunzioni di partenza, la tesi è ciò che si vuole dimostrare essere conseguenza logica delle ipotesi e dei postulati. L'insieme dei ragionamenti logici che dalle ipotesi permettono di giungere alla tesi prende il nome di **dimostrazione** del teorema.
Come esempio completo di utilizzo di Dr. Geo e per chiarire i concetti appena introdotti analizziamo un curioso teorema dovuto ad Aubel.

Teorema 1 *Sia $ABCD$ un quadrilatero convesso e i punti L, M, N e O siano i centri dei quadrati costruiti sui lati del quadrilatero come in figura 1.2. Allora:*

1. *i segmenti LN e MO hanno la stessa lunghezza;*
2. *i segmenti LN e MO sono perpendicolari.*

Dal momento che Dr. Geo non implementa la costruzione di un quadrato realizziamo innanzitutto una macro-costruzione che dati gli estremi di un segmento AB permetta di disegnare il quadrato di lato AB e il suo centro. Salviamo questa macro `quadratocentro.mgeo` e ricorrendo ad essa riproduciamo la figura relativa al teorema di Aubel.
Le ipotesi del teorema sono:

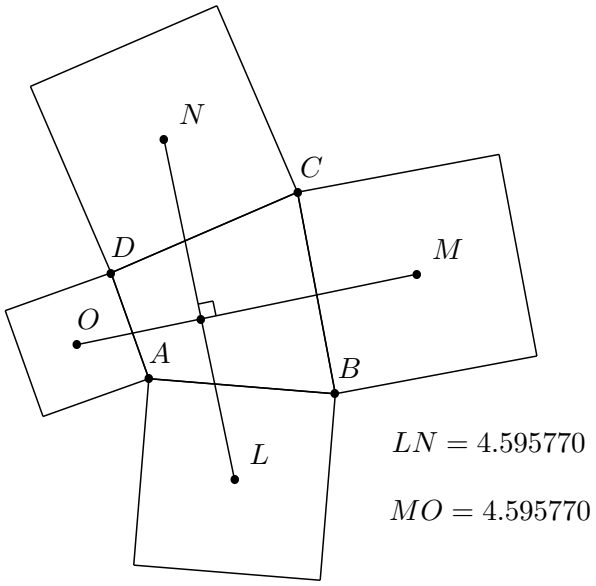


Figura 1.2: Teorema di Aubel

- $ABCD$ è un quadrilatero convesso
- L, M, N, O sono i centri dei quadrati costruiti sui lati del quadrilatero.

La tesi del teorema è che:

- $LN = MO$
- $LN \perp MO$ che si legge "LN è perpendicolare a MO".

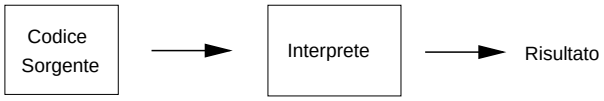
In questo contesto non siamo interessati alla dimostrazione (non immediata) del teorema ma a comprendere il significato geometrico del suo enunciato. Osserviamo subito, anche se su questo aspetto torneremo nel seguito, che con Dr. Geo **non si possono dimostrare teoremi** in quanto lo strumento informatico non è in grado di effettuare dei ragionamenti. L'ambiente di geometria dinamica ci è tuttavia utile per penetrare più in profondità il significato dell'enunciato del teorema e, attraverso lo strumento di movimento, per analizzarne efficacemente diversi casi particolari.
Utilizzando righello e goniometro di Dr. Geo misuriamo le lunghezze dei segmenti LN e MO e l'angolo che essi formano. Deformando dinamicamente il quadrilatero noteremo che in effetti quanto previsto dal teorema trova sempre perfetta corrispondenza.

Esercizio 3 *Cosa accade se il quadrilatero non è più convesso?*

Capitolo 2

Script Guile per Dr. Geo

Il linguaggio di programmazione di cui imparerai i fondamenti è Scheme. Scheme è un **linguaggio di alto livello**. Altri linguaggi di alto livello di cui forse hai sentito parlare sono il Pascal, il C e Java.
Come avrai intuito sentendo l'espressione linguaggio di alto livello esistono anche **linguaggi di basso livello**, talvolta chiamati linguaggi macchina. In modo non completamente corretto possiamo dire che i computer possono eseguire solamente programmi scritti in linguaggio di basso livello: i programmi scritti con linguaggi di alto livello devono necessariamente essere elaborati prima di essere eseguiti.
I programmi scritti in linguaggio di alto livello sono generalmente più facili da scrivere rispetto a quelli in linguaggio di basso livello, in quanto i linguaggi di alto livello sono stati progettati con lo scopo esplicito di risultare facilmente leggibili e utilizzabili dagli esseri umani.
Uno dei modi con cui i programmi di alto livello vengono trasformati in programmi di basso livello prende il nome di **interpretazione**. Un interprete legge il programma e lo esegue trasformando ogni riga di istruzioni in una azione. L'interprete elabora il programma un po' alla volta, alternando la lettura delle istruzioni all'esecuzione dei comandi che le istruzioni descrivono:



In Dr. Geo è disponibile l'interprete Guile del linguaggio Scheme. L'utilizzatore di Dr. Geo può quindi eseguire degli **script** in linguaggio Scheme che hanno in ingresso (input) un sottoinsieme (anche vuoto) di oggetti geometrici e che restituiscono in uscita (output) un numero o una stringa. Gli esempi che seguono hanno lo scopo di chiarire il modo peculiare in cui Guile viene utilizzato in Dr. Geo.

2.1 Aritmetica Scheme

Il primo gruppo di esempi che analizziamo mostra come gli script Guile permettano di disporre in Dr. Geo di una calcolatrice. In generale i simboli speciali che rappresentano elaborazioni di tipo matematico vengono chiamati in programmazione **operatori**. I valori che l'operatore usa nei calcoli sono detti **operandi**. Le operazioni fondamentali che consideriamo inizialmente sono l'addizione `+`, la sottrazione `-`, la moltiplicazione `*` e la divisione `/`.
Il primo esempio che consideriamo consiste nell'eseguire in Dr. Geo il prodotto tra numeri naturali:

$$20 \cdot 32$$

Il procedimento per effettuare il calcolo è il seguente:

- si seleziona il comando di Dr. Geo che attiva uno script Guile;
- si clicca con il mouse direttamente sullo sfondo del foglio di lavoro nel punto in cui si desidera che compaia il valore di uscita dello script (nel nostro caso il risultato dell'operazione). Una volta collocato lo script apparirà la scritta celeste Dr. Genius;
- si clicca sul comando per modificare le proprietà di un oggetto (icona a chiave inglese) e cliccando successivamente sulla scritta Dr. Genius si aprirà una finestra di dialogo;
- a questo punto si cancella tutto il contenuto dello script e lo si sostituisce con l'espressione Scheme `(* 20 32)`
- finalmente si clicca su Applica o su Ok e si ottiene il risultato 640.00.

L'esempio precedente è istruttivo sotto diversi punti di vista:

1. l'implementazione delle operazioni in Scheme, diversamente da come siamo abituati, avviene scrivendo prima l'operatore e successivamente gli operandi su cui l'operatore agisce. Nell'esempio precedente prima viene posto l'operatore di moltiplicazione cui seguono i due operandi 20 e 32;
2. il linguaggio Scheme, come vedremo anche nel seguito, utilizza abbondantemente le parentesi. Nell'esempio precedente l'espressione Scheme che implementa il calcolo del prodotto è stata racchiusa tra parentesi tonde.

Oltre a questo, osserviamo che il risultato dell'operazione viene rappresentato in forma decimale con un'approssimazione di due cifre dopo la virgola. Il numero di cifre dell'approssimazione può essere controllato modificando le Preferenze di Dr. Geo [2].

Un secondo esempio di script, leggermente più complesso del precedente, permette di calcolare il risultato dell'espressione aritmetica:

$$(20 \cdot (32 + 2) - 15) : 2$$

Tecnicamente la procedura da seguire per ottenere il risultato in Dr. Geo è la stessa vista nell'esempio precedente. L'unica differenza è rappresentata dall'espressione Scheme che serve per implementare l'espressione aritmetica:

```
( / ( - (* 20 (+ 32 2)) 15) 2 )
```

In questa espressione compaiono più operatori e quindi assume un'importanza cruciale l'ordine in cui le diverse espressioni vengono valutate. Le **regole di precedenza** vengono gestite in Scheme attraverso l'annidamento di parentesi, in modo del tutto simile a quanto accade in matematica.

Esercizio 4 *Calcola con Dr. Geo il risultato di* $2 : (3 - 12.5) + 3.4$ *e confrontalo con quello di una calcolatrice tascabile.*

2.2 Funzioni matematiche

Un ulteriore esempio di implementazione di un calcolo matematico consiste nel valutare la seguente espressione:

$$\sqrt{11^2 + 23}$$

Lo script che permette di eseguire il calcolo è

```
(sqrt (+ ( expt 11 2) 23))
```

ed osserviamo che in esso compaiono le **funzioni**:

- `(sqrt x)` (square root) che calcola la radice quadrata $\sqrt{x}$;
- `(expt x n)` (exponentiation) che calcola la potenza x^n.

Scheme implementa molte altre funzioni il cui elenco completo può essere reperito in un qualsiasi manuale dedicato a questo linguaggio [7].

Esercizio 5 *Calcola con Dr. Geo il valore dell'espressione* $\sqrt{4^2 + 7^2}$ *e confrontalo con quello di una calcolatrice tascabile.*

2.3 Soluzione di problemi

Un utilizzo abbastanza immediato degli strumenti di calcolo messi a disposizione da Guile consiste nell'eseguire i calcoli che permettono di risolvere un problema. Analizziamo in dettaglio il seguente:

Problema *Siano date due circonferenze concentriche di raggio rispettivamente* $r_1 = 4$ *e* $r_2 = 2$ *centimetri. Calcolare l'area della corona circolare che esse delimitano.*

Ricorrendo alle funzionalità base di Dr. Geo possiamo in primo luogo disegnare la figura relativa a questo problema. Utilizziamo quindi uno script Guile per risolvere concretamente il problema. In questo caso specifico lo script che dobbiamo scrivere ha bisogno di disporre in input dei valori numerici dei raggi r_1 e r_2 delle due circonferenze. Per questa ragione la procedura da seguire in Dr. Geo è leggermente diversa da quella vista in precedenza per gli script senza oggetti geometrici in ingresso:

- si seleziona il comando che attiva uno script Guile;
- si selezionano ordinatamente le circonferenze di raggio r_1 e r_2;
- si clicca con il mouse direttamente sullo sfondo del foglio di lavoro nel punto in cui si desidera che compaia il valore di uscita dello script che nel nostro caso sarà il risultato dell'operazione. Una volta collocato lo script apparirà la scritta celeste Dr. Genius;
- si clicca sul comando per modificare le proprietà di un oggetto (icona a chiave inglese) e cliccando successivamente sulla scritta Dr. Genius si apre una finestra di dialogo;
- a questo punto si cancella il contenuto dello script e lo si sostituisce con il codice:

```
(define r1 (getRadius a1))
(define r2 (getRadius a2))
(* 3.1415 (- (* r1 r1)(* r2 r2)))
```

- finalmente si clicca su Ok e si ottiene il risultato 37.70.

Lo script che permette di risolvere il problema contiene diversi aspetti nuovi. Essi sono tutti contenuti concettualmente nella prima riga:

```
(define r1 (getRadius a1))
```

Le espressioni Scheme del tipo:

```
(define variabile valore)
```

permettono di assegnare ad una variabile un valore. Nel nostro esempio la **variabile** è `r1` e ad essa viene assegnato il valore r_1 che esprime il raggio della prima circonferenza.
Il valore da assegnare ad una variabile viene gestito nello Scheme di Dr. Geo da espressioni del tipo:

```
(metodo oggetto)
```

Con riferimento all'esempio precedente, il **metodo di Dr. Geo** `getRadius` ha come argomento l'oggetto geometrico `a1` che, per definizione, è il primo oggetto su cui si è cliccato con il mouse, ossia la prima circonferenza selezionata. In modo del tutto analogo nella seconda riga di codice viene definita la **variabile** `r2` a cui viene assegnato il valore r_2. Il metodo `getRadius` in questo caso si riferisce all'oggetto `a2` che è il secondo oggetto selezionato con il mouse ossia la circonferenza di raggio r_2.

> *L'elenco completo dei metodi per Dr. Geo è contenuto nell'appendice A*

Esercizio 6 *Sia dato un trapezio rettangolo $ABCD$ di cui sono note le lunghezze delle due basi e dell'altezza. Calcolare perimetro e area del trapezio.*

2.4 Teoremi e Congetture

Gli ambienti per lo studio della geometria dinamica non possono essere utilizzati come strumenti per dimostrare teoremi. Tuttavia essi possono essere utilizzati in modo proficuo per almeno tre tipi di indagine:

1. comprendere l'importanza delle ipotesi negli enunciati dei teoremi;

2. stabilire se una congettura può essere candidata a divenire un enunciato di un teorema;

3. falsificare una congettura.

Una **congettura** è una proposizione di cui *non* è disponibile la dimostrazione. Il destino di una congettura può essere duplice: la congettura diviene un teorema nel momento in cui si scopre come dimostrarla, la congettura viene **falsificata** trovando un controesempio. Osserviamo che è sufficiente un *unico* controesempio per falsificare una congettura.

2.4.1 Teorema di Tolomeo

L'esempio che trattiamo, riportato anche in [2], è particolarmente interessante e si può ritrovare nella sua formulazione originale in [14].

Teorema 2 *(di Tolomeo) Dato un qualsiasi quadrilatero convesso inscritto in una circonferenza, la somma dei prodotti dei lati opposti è uguale al prodotto delle diagonali.*

Questo teorema contiene un'ipotesi che spesso si tende a tralasciare in quanto apparentemente inutile per la sua dimostrazione. Si tratta dell'ipotesi che il quadrilatero sia convesso. Non è nostra intenzione ripercorrere l'intera dimostrazione del teorema ma vogliamo mettere in evidenza come utilizzando Dr. Geo sia possibile comprendere l'importanza di questa ipotesi. Con riferimento alla figura 2.1 possiamo riformulare la tesi del teorema di Tolomeo per il quadrilatero $ABCD$ scrivendo:

$$AC \cdot BD = AB \cdot DC + BC \cdot DA$$

possiamo quindi scrivere uno script Guile che ha in input i lati del quadrilatero di figura e che restituisce in output la somma dei prodotti dei lati opposti di $ABCD$:

```
(define AB (getLength a1))
(define DC (getLength a2))
(define BC (getLength a3))
(define AD (getLength a4))
(+ (* AB DC )(* BC AD ))
```

Il significato dello script è intuitivo: notiamo solo la presenza del metodo per Dr. Geo `getLength` che permette di ricavare la lunghezza di un oggetto geometrico.
Possiamo successivamente scrivere un secondo script, che ha in input le due diagonali del quadrilatero di figura e che restituisce in output il loro prodotto:

```
(define DB (getLength a1))
(define AC (getLength a2))
(* DB AC )
```

Deformando dinamicamente la figura noteremo che i risultati dei due script restano uguali ad eccezione del caso in cui il quadrilatero perde la convessità!

Esercizio 7 *Utilizzando Dr. Geo realizza una figura che ponga in evidenza l'importanza delle ipotesi contenute nel teorema di Pitagora.*

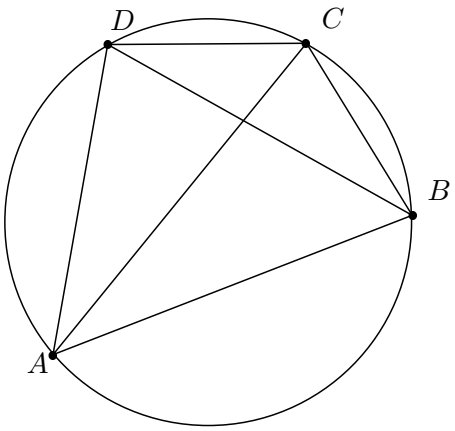


Figura 2.1: Teorema di Tolomeo

2.4.2 Angoli interni e quadrilateri

Come sappiamo il rettangolo è un particolare quadrilatero e la somma dei suoi angoli interni è pari a 360°. La congettura che avanziamo è la seguente:

Congettura *La somma degli angoli interni di un qualsiasi quadrilatero vale 360°.*

Possiamo verificare se la congettura può essere candidata a divenire l'enunciato di un teorema con Dr. Geo procedendo come segue:

- disegniamo in Dr. Geo un quadrilatero qualsiasi
- utilizzando il goniometro misuriamo i suoi angoli interni
- scriviamo uno script Guile che ha in input gli angoli interni e che restituisce la loro somma:

```
(define angolo1 (getAngle a1))
(define angolo2 (getAngle a2))
(define angolo3 (getAngle a3))
(define angolo4 (getAngle a4))
(+ angolo1 angolo2 angolo3 angolo4)
```

- deformando dinamicamente la figura noteremo che gli angoli interni cambiano mentre l'uscita dello script rimane sempre pari a 360°.

Non è difficile vedere che questa congettura è in realtà un teorema che si dimostra facilmente.

Dimostrazione. Una diagonale divide un qualsiasi quadrilatero in due triangoli. Sapendo che la somma degli angoli interni di un triangolo è 180° è facile concludere la dimostrazione. □

Da un punto di vista della programmazione, osserviamo che lo script che permette di calcolare la somma degli angoli interni del triangolo equilatero utilizza il metodo per Dr. Geo `getAngle` che permette di ricavare l'ampiezza in gradi di un angolo.

2.4.3 Controesempi

Consideriamo la seguente congettura che nasce da un modo errato di generalizzare il teorema di Pitagora.

Congettura *La somma dei quadrati delle lunghezze di due lati di un triangolo è uguale al quadrato del lato di lunghezza maggiore.*

Per mostrare che questa congettura è falsa basta riprodurre la figura in Dr. Geo e scrivere due script Guile: il primo calcola la somma dei quadrati dei primi due lati e il secondo il quadrato del lato maggiore. Si noterà che i valori restituiti dagli script sono diversi a parte nel caso in cui il triangolo sia rettangolo. Un esempio che mostra che una congettura è errata prende il nome di **controesempio**. Ribadiamo che in Matematica una congettura è falsificata una volta trovato un *solo* controesempio!

2.5 Espressioni condizionali

Un'**espressione booleana** è un'espressione che è *vera* o *falsa*. Per fare un esempio consideriamo i valori S_1 e S_2 delle aree di due triangoli qualsiasi e scriviamo la seguente espressione:

$$S_1 \neq S_2$$

che si legge "S_1 è diversa da S_2". Non è difficile comprendere che si tratta di un'espressione booleana in quanto dati due triangoli o essi hanno la stessa area oppure l'area del primo è diversa da quella del secondo.
Partendo da un'espressione booleana possiamo definire due tipi di espressioni **condizionali**:

1. `(if condizione istruzione1 istruzione2)`: se (`if`) l'espressione `condizione` è vera viene eseguita `istruzione1` altrimenti, se è falsa, viene eseguita `istruzione2`.

2. `(if condizione istruzione)`: se (`if`) l'espressione `condizione` è vera viene eseguita `istruzione`.

Vediamo di chiarire con un semplice esempio come possano essere sfruttate le espressioni condizionali.

2.5.1 Problema di massimo

In matematica sappiamo che il triangolo inscritto in una circonferenza di area massima è il triangolo equilatero. Possiamo analizzare la *plausibilità* di questa ipotesi utilizzando gli script Guile ricorrendo a questa strategia:

- costruiamo un triangolo equilatero e la circonferenza in cui esso è inscritto;
- disegniamo un triangolo qualsiasi inscritto nella circonferenza;
- con due script calcoliamo l'area del triangolo equilatero S_m e del triangolo generico S;
- a questo punto scriviamo uno script che ha in input i valori di S_m e di S e che restituisce la **stringa** `Vero` se $S \leq S_m$ e `Attenzione: falso!` altrimenti:

  ```
  (define sm (getValue a1))
  (define s  (getValue a2))
  ((if >= sm s) "Vero" "Attenzione: falso!")
  ```

  Osserviamo che nello script è stato utilizzato il metodo di riferimento `getValue` che assegna alla variabile `sm` il valore `a1` che coincide con il numero S_m che rappresenta l'uscita di uno script.

- deformando dinamicamente il triangolo generico in qualsiasi modo, si noterà che la segnalazione di falso *non compare mai*.

Ribadiamo con forza che il procedimento precedente **non** ha alcun valore dimostrativo. Il suo unico scopo è quello di assicurarci della plausibilità di un enunciato.

Esercizio 8 *Disegna con Dr. Geo un triangolo ABC di base AB. L'asse della base divide il piano in due semipiani α e β. Scrivi uno script Guile che stampi sullo schermo* `Beta` *quando il vertice C del triangolo è interno al semipiano β.*

Capitolo 3

Figure Scheme per Dr. Geo

Le Figure Scheme sono figure scritte in un linguaggio relativamente naturale e sono compatibili con il linguaggio Scheme. Da un punto di vista concettuale, non si tratta più di costruire una figura utilizzando gli strumenti messi a disposizione dall'interfaccia grafica ma piuttosto di descrivere una figura utilizzando il linguaggio di programmazione Scheme. Scheme è per sua natura un linguaggio di alto livello e per questa ragione, una volta definita una figura, avremo a disposizione tutta la sua potenza per eseguire azioni ricorsive su parti della figura o per utilizzare funzioni aleatorie per modificarla ad ogni sua comparsa sul foglio di lavoro. In questo modo le Figure Scheme sono allo stesso tempo svincolate dall'interfaccia grafica e potenziate dalla presenza di Scheme.
Da un punto di vista tecnico, le Figure Scheme sono dei file scritti in linguaggio Scheme che contenengono un certo numero di **macro** e che, una volta valutati da Dr. Geo, restituiscono una figura geometrica. Nel seguito, per brevità, indicheremo talvolta le Figure Scheme per Dr. Geo con l'acronimo FSD.

3.1 Creare una Figura Scheme

Prima di procedere all'analisi di un buon numero di esempi di FSD, ricordiamo la procedura generale che serve per creare una figura Scheme e per visualizzarla. Le figure Scheme sono a tutti gli effetti dei programmi il cui **codice sorgente** viene scritto con un editor di testo.
Quando si parla di editor di testo il riferimento immediato è rappresentato dagli editor di testo che usiamo quotidianamente per scrivere relazioni, lettere, tesine e via di seguito. Questi editor, così utili per creare gradevoli impaginazioni, per creare titoli accattivanti, per inserire immagini e molto altro, in generale **non sono adatti** a scrivere programmi. Alcuni di essi non permettono neppure di salvare un file nel **formato** Scheme di cui ci serviremo sistematicamente: in altri termini alcuni editor non per-

mettono di salvare un file con nome del tipo `nomefile.scm`, caratterizzato dall'**estensione** `scm` che indica che il file contiene codice scritto in Scheme. Oltre a questo, gli editor di testo che usiamo di solito, non essendo progettati per scrivere codice, mancano di funzionalità molto utili in programmazione come ad esempio la gestione delle parentesi.

Ci sono diversi editor che possiamo consigliarti per scrivere il codice di una figura Scheme (Scite, Vi, Emacs) tuttavia nel nostro corso useremo Kwrite con la modalità di evidenziazione per Scheme.

Supponiamo ora di aver creato una Figura Scheme e dì averla salvata con nome `figura.scm`. L'**output** di `figura.scm` sarà rappresentato da una figura geometrica visualizzabile in un foglio di lavoro di Dr. Geo.

Per poter eseguire una Figura Scheme e per visualizzare il suo output è sufficiente aprire Dr. Geo, azionare il comando File → Valuta *e selezionare il file, ad esempio* `figura.scm`*, che individua la figura.*

3.2 Esempi base

Iniziamo studiando alcuni esempi di Figure Scheme facilmente comprensibili. Dopo aver aperto l'editor Kwrite scrivi le seguenti righe:

```
(new-figure "Figura")
(lets Point "A" free 2 -2)
```

Salva il file con nome `prova.scm` in una cartella. Apri Dr. Geo e utilizzando il comando File → Valuta apri il file `prova.scm`. Sul foglio di lavoro di Dr. Geo comparirà una figura con nome *Figura* che contiene il solo punto A di coordinate cartesiane $(2, -2)$.
La prima riga di codice ha il solo scopo di aprire un foglio di lavoro in Dr. Geo e di assegnare alla figura, inizialmente priva di oggetti, il nome *Figura*.
La seconda riga permette di definire un oggetto geometrico attraverso un'espressione Scheme. La sintassi di questa espressione deve essere compresa a fondo in quanto essa rappresenta il prototipo per la definizione di ogni altro oggetto:

- l'espressione inizia con `lets`;
- segue l'**oggetto** che verrà costruito che in questo caso è `Point` (punto);
- quindi tra virgolette compare il **simbolo** dell'oggetto, nel nostro caso `A`. Se non si desidera attribuire alcun simbolo all'oggetto è sufficiente scrivere le virgolette senza precisare alcun contenuto;
- infine si definiscono le **proprietà** dell'oggetto: il tipo di punto `free` (libero) e le sue coordinate.

Le espressioni Scheme necessarie per costruire tutti gli oggetti geometrici di Dr. Geo sono riportate nell'appendice B.

Il secondo esempio che discutiamo è utile per imparare a modificare gli **attributi** degli oggetti geometrici che vengono costruiti:

```
(new-figure "Send")

(lets Point "A" free 1 0)
(lets Point "B" free 2 0)
(lets Point "C" free 2 0)
(lets Line "d1" 2points A B)

(send A color yellow)
(send A shape round)
(send A size large)
(send B color green)
(send C masked)
(send d1 thickness dashed)
```

Prima di passare all'analisi della Figura Scheme è fondamentale valutarla in modo da confrontare il testo del codice sorgente con quanto compare sul foglio di lavoro di Dr. Geo!
Le prime quattro espressioni permettono di costruire tre punti e una retta. La parte di codice che qui maggiormente ci interessa riguarda la serie di comandi che iniziano con `send`. Attraverso `send` possiamo comunicare con un oggetto di cui sia disponibile il relativo **simbolo**. Nel caso dell'esempio precedente i simboli di cui disponiamo sono `A`, `B`, `C` e `d1` e vengono scritti *senza* utilizzare le virgolette. La sintassi di `send` è del tipo:

```
(send simbolo attributo modo)
```

Il simbolo identifica l'oggetto con cui si vuole comunicare, l'attributo definisce la caratteristica del simbolo cui è rivolta la comunicazione, il modo specifica una modalità per l'attributo. Ad esempio `(send A color yellow)` invia al simbolo `A` (punto A) un messaggio che riguarda l'attributo `color` (colore) specificando il modo `yellow` (giallo): molto semplicemente il punto A viene disegnato sul foglio di lavoro con colore giallo. Il significato dei rimanenti comandi dovrebbe a questo punto risultare chiaro: il punto A ha `shape` (forma) `round` (circolare), il punto A ha `size` (dimensione) `large` (grande), il punto B ha `color` (colore) `green` (verde), il punto C è `masked` (nascosto), la linea d_1 ha `thickness` (spessore) `dashed` (tratteggiato).

3.2.1 Figure casuali

L'esempio di Figura Scheme che studiamo in questo paragrafo ha il seguente codice:

```
(new-figure "Random")

(define (triangolo p1 p2 p3)

  (Segment "" extremities p1 p2)
  (Segment "" extremities p2 p3)
  (Segment "" extremities p1 p3)
)

(define (rand) (- 8 (* 16 (random:uniform))))

(lets Point "A" free (rand) 0)
(lets Point "B" free 5 0)
(lets Point "C" free (rand) 5)

(triangolo A B C)
```

Se si valuta la figura con Dr. Geo si scoprirà che l'output è rappresentato da un triangolo ABC. Se si valuta una seconda volta la stessa figura si noterà che il triangolo che compare sul foglio di lavoro è diverso dal primo.
L'esempio è di notevole interesse in quanto ci mostra come, a partire dagli oggetti disponibili in Dr. Geo, si possano costruire oggetti più complessi. Il primo oggetto che compare nel codice è un triangolo che viene definito attraverso la seguente **funzione** Scheme:

```
(define (triangolo p1 p2 p3)

  (Segment "" extremities p1 p2)
  (Segment "" extremities p2 p3)
  (Segment "" extremities p1 p3)
)
```

La sintassi per definire una funzione è del tipo:

```
(define (nome oggetti_base)(oggetto1)(oggetto2)...)
```

Abbiamo già incontrato `define` a proposito degli Script Guile. La prima coppia di parentesi tonde contiene il nome della funzione e l'elenco degli oggetti geometrici da cui essa dipende. Nel caso della funzione `triangolo` gli oggetti da cui essa dipende sono i tre vertici del triangolo. Seguono i diversi oggetti che permettono di costruire un nuovo oggetto. Sempre nel caso della funzione `triangolo` gli oggetti sono i tre segmenti che individuano i lati del triangolo.
Il secondo oggetto è un numero **casuale** che viene definito attraverso la funzione `rand`:

```
(define (rand) (- 8 (* 16 (random:uniform))))
```

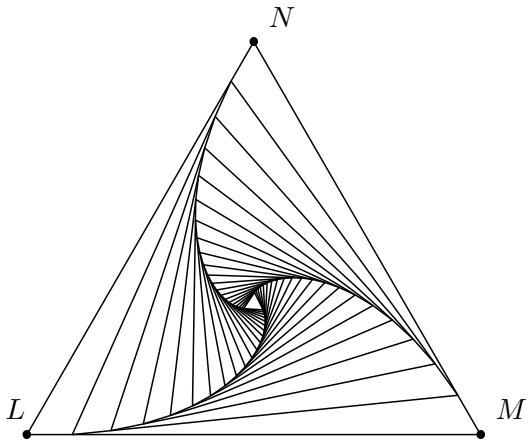


Figura 3.1: Figura ricorsiva

La funzione `rand` non dipende da nessun oggetto di Dr. Geo ma ritorna un numero casuale compreso tra -8 e 8. L'espressione `(random:uniform)` genera un numero casuale nell'intervallo $[0, 1]$, moltiplicando questo numero per 16 si ottiene un numero casuale nell'intervallo $[0, 16]$, sottraendo 8 si ottiene un numero casuale nell'intervallo $[-8, 8]$.

3.3 Funzioni ricorsive

Iniziamo da un esempio di FSD che ha come output la figura 3.1.

Il codice Scheme, con i relativi commenti, che permette di realizzarla è il seguente:

```
(new-figure "Triangoli")

(define (triangolo p1 p2 p3 n)

    ;; costruzione geometrica iterativa

   ( let* (
            (s1 (Segment "" extremities p1 p2))
            (s2 (Segment "" extremities p2 p3))
            (s3 (Segment "" extremities p3 p1))
            (A  (Point   "" on-curve s1  1/10))
            (B  (Point   "" on-curve s2  1/10))
            (C  (Point   "" on-curve s3  1/10)))
```

```
   ;; nascondiamo alcuni oggetti geometrici

(send A masked)
(send B masked)
(send C masked)

     ( if (> n 0)

       (triangolo A B C (- n 1)))
    )
)

  ;; triangolo di partenza

(lets Point "L" free -5 0)
(lets Point "M" free  5 0)
(lets Point "N" free  0 ( * 5 (sqrt 3)))

(triangolo L M N 20)  ;; numero di iterazioni 20
```

I commenti, che non vengono considerati quando il programma viene eseguito, sono anticipati da un doppio punto e virgola. Procediamo quindi all'analisi dettagliata del codice. Osserviamo di passaggio che lo studio di codice scritto da altri è una prassi essenziale sia per l'apprendimento della programmazione che per porre le basi dello sviluppo di nuovi esempi utilizzando il metodo dell'analogia.

3.3.1 Analisi del codice

Il codice precedente può essere idealmente suddiviso in tre sezioni che chiamiamo: *titolo*, *funzione ricorsiva* e *oggetto iniziale*.

- *Titolo*: il titolo è rappresentato solo dalla prima linea di codice in cui attraverso l'istruzione `new-figure` viene dato il nome *Triangoli* alla figura Scheme che si vuole costruire.
- *Funzione ricorsiva*: questa è la parte più importante del codice. La funzione `triangolo` dipende da tre punti geometrici (simboli `p1`, `p2` e `p3`) e da un numero naturale (simbolo `n`). Il costrutto `let*` viene utilizzato per definire una serie di oggetti geometrici. Tramite `send` alcuni oggetti geometrici vengono mascherati. Finalmente un **ciclo** gestito dall'istruzione condizionale `if` permette la ripetizione della costruzione.
- *Oggetto iniziale*: l'ultima parte del codice permette di definire l'oggetto geometrico di partenza dell'intera costruzione ossia il triangolo

LMN. Notiamo ancora una volta il modo funzionale in cui le funzioni matematiche vengono implementate in Scheme per cui l'operazione di moltiplicazione $5\sqrt{3}$ viene implementata nella forma `(* 5 (sqrt 3))`.

Esercizio 9 *Modifica il codice precedente in modo che la successione dei triangoli sia di colore blu e con i lati tratteggiati.*

Esercizio 10 *Modifica il codice precedente in modo da ottenere una successione di quadrati.*

3.3.2 Spirale di Baravelle

Iniziamo considerando il seguente codice:

```
(new-figure "Baravelle")

(define (quad p1 p2 p3 p4 n)

     ( let* (
              (s1 (Segment "" extremities p1 p2))
              (s2 (Segment "" extremities p2 p3))
              (s3 (Segment "" extremities p3 p4))
              (s4 (Segment "" extremities p4 p1))
              (A  (Point   ""   on-curve s1 1/2))
              (B  (Point   ""   on-curve s2 1/2))
              (C  (Point   ""   on-curve s3 1/2))
              (D  (Point   ""   on-curve s4 1/2)))

      ( if (> n 0)

        (quad A B C D (- n 1)))
     )
)

(lets Point "A" free  5  5)
(lets Point "B" free -5  5)
(lets Point "C" free -5 -5)
(lets Point "D" free  5 -5)

(quad A B C D 7)
```

Non è difficile comprendere che la figura generata è una successione di quadrati. Il primo passo della costruzione richiama la costruzione descritta dal filosofo Platone nel *Menone* [11] a proposito dell'importantissimo problema classico della duplicazione del quadrato. Ricordiamo che il problema

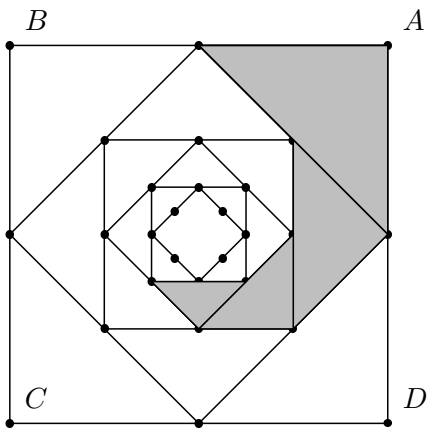


Figura 3.2: Spirale di Baravelle

consisteva nel costruire, a partire da un quadrato di area fissata, un quadrato di area doppia. Ritornando al codice precedente la figura ad esso associata è la figura 3.2. In essa abbiamo successivamente evidenziato in grigio alcuni termini della successione di triangoli T_n che definisce una *Spirale di Baravelle*. La spirale di Baravelle è un esempio rilevante di figura ricorsiva in quanto permette di introdurre geometricamente il concetto di *progressione geometrica*. Supponiamo che l'area del quadrato $ABCD$ sia unitaria. Allora l'area A_0 del primo triangolo T_0 della spirale è pari a:

$$A_0 = \frac{1}{8}$$

l'area A_1 del secondo triangolo T_1 è pari a:

$$A_0 = \frac{1}{16}$$

e così di seguito. La somma A_n delle aree dei primi n triangoli della spirale è pari a:

$$A_n = \frac{1}{8} \cdot \frac{1 - 2^{-n-1}}{1 - 2^{-1}}$$

Per n "molto grande", il termine 2^{-n-1} è molto piccolo e quindi l'area della spirale vale:

$$A = \frac{1}{4}$$

come per altro si può intuire semplicemente guardando la figura.
Ritornando alla programmazione consideriamo il seguente esercizio:

Esercizio 11 *Disegna con carta e penna il tipo di figura che compare sul foglio di lavoro quando Dr. Geo valuta la seguente Figura Scheme:*

```
(new-figure "Who")

(define (triangle p1 p2 p3 n)

 (let* (
  (s1 (Segment "" extremities p1 p2))
  (s2 (Segment "" extremities p2 p3))
  (s3 (Segment "" extremities p3 p1))
  (m  (Point   "" middle-2pts p1 p3))
  (r  (Segment "" extremities m p3))
  (pe (Line    "" orthogonal p3 s3))
  (ci (Circle  "" center-segment p3 r))
  (p4 (Point   "" intersection2 pe ci)))

  (send pe masked)
  (send ci masked)
  (send p4 masked)
  (send m  masked)

    (if (> n 0)

     (triangle m p3 p4 (- n 1)))
   )
)

(lets Point "A" free 0 5)
(lets Point "B" free 5 5)
(lets Point "C" free 5 0)

(triangle A B C 9)
```

Verifica la tua risposta valutando il codice Scheme in Dr. Geo.

3.3.3 Spirale dei numeri irrazionali

Un famoso esempio di figura ricorsiva è la *spirale di Teodoro*. Si tratta di una costruzione geometrica antica [15] che permette di rappresentare le radici quadrate dei numeri naturali a partire da un triangolo rettangolo isoscele con cateti di lunghezza unitaria.
Consideriamo un triangolo rettangolo isoscele T_0 con i cateti di lunghezza unitaria. Per il teorema di Pitagora si ha allora che l'ipotenusa ha lunghezza pari a $\sqrt{2}$. Se ora si costruisce un nuovo triangolo rettangolo T_1 che ha un cateto unitario e il secondo cateto coincidente con l'ipotenusa di T_0, sempre per il teorema di Pitagora, è chiaro che l'ipotenusa di T_1 ha lunghezza $\sqrt{3}$. Iterando il procedimento si ottengono tutte le radici quadrate dei numeri

naturali.
Dal numero irrazionale $3\sqrt{2}$ in poi i triangoli che formano la spirale cominciano purtroppo a sovrapporsi. Per questo motivo la costruzione geometrica viene considerata una rappresentazione delle radici quadrate dei numeri da 2 a 17 e la tradizione vuole che proprio questa figura sia stata tracciata dal matematico greco Teodoro sulla sabbia in un episodio ricordato da Platone nel dialogo il *Teeteto* [11] dove leggiamo:

Teodoro, qui, stava tracciando una figura sulle potenze, quella di tre piedi e quella di cinque, mostrando che esse, quanto alla lunghezza, non sono commensurabili con la misura di unità di un piede; e così scegliendole una per una fino a quella di diciassette piedi: a questa, non so perché, si è fermato.

Possiamo tradurre la costruzione precedente in Figura Scheme:

```
(new-figure "Spirale di Teodoro")

(define (triangle p1 p2 p3 n)

   (let* (
     (s1 (Segment "" extremities p1 p2))
     (s2 (Segment "" extremities p2 p3))
     (s3 (Segment "" extremities p3 p1))
     (pe (Line    "" orthogonal p3 s3))
     (ci (Circle  "" center-segment p3 s2))
     (p4 (Point   "" intersection2 pe ci)))

     (send pe masked)
     (send ci masked)
     (send p4 masked)

      (if (> n 0)
        (triangle p1 p3 p4 (- n 1)))
    )
)

(lets Point "O" free  0 0)
(lets Point "A" free -1 0)
(lets Point "B" free -1 1)

(triangle O A B 15)
```

Una volta valutato da Dr. Geo il codice restituisce la spirale di figura 3.3. L'ipotenusa di ogni triangolo ha lunghezza uguale alla radice quadrata di un numero naturale compreso tra 2 e 17.

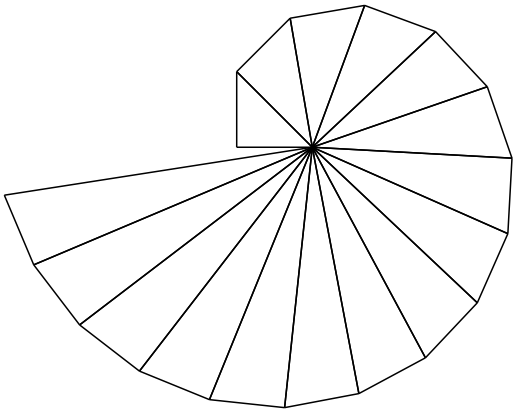


Figura 3.3: Spirale di Teodoro

3.3.4 Tassellazioni per sostituzione

In modo del tutto informale e restrittivo, possiamo dire che una **tassellazione** del piano è un ricoprimento con poligoni che hanno al più i lati in comune. Un pavimento di piastrelle quadrate è un esempio banale di tassellazione del piano con poligoni quadrati. In questo paragrafo desideriamo soffermarci su un tipo particolare di tassellazioni note come **tassellazioni per sostituzione**.
Lo studio delle tassellazioni per sostituzione ha ricevuto notevole attenzione nell'ambito della matematica moderna anche per questioni non riguardanti strettamente la geometria. Oltre a questo, molte tassellazioni per sostituzione possono essere modificate, imponendo opportune regole di composizione, per trasformarle in **tassellazioni aperiodiche** ossia in tassellazioni in cui, *comunque vengano disposti* i tasselli di base, la tassellazione che si ottiene non ammette simmetrie di traslazione.
In questo contesto, per familiarizzare con l'argomento, vogliamo studiare con Dr. Geo l'esempio di una tassellazione per sostituzione inventata dal matematico John Conway e nota come *pinwheel tiling* (tassellazione del mulino a vento) [13]. L'osservazione elementare da cui Conway prende le mosse riguarda la possibilità di decomporre un triangolo rettangolo, con cateti l'uno doppio dell'altro, in cinque triangoli congruenti e simili a quello di partenza (figura 3.4).

La riproduzione della decomposizione si può realizzare facilmente con Dr. Geo ricorrendo alle funzionalità di base (retta perpendicolare, punto medio, circonferenza).

Esercizio 12 *Riproduci con Dr. Geo la costruzione che rappresenta la*

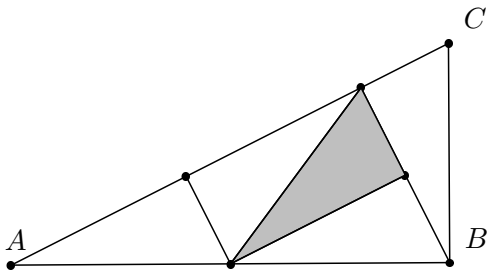


Figura 3.4: Costruzione di Conway

decomposizione di Conway.

Per estendere questa costruzione ad una tassellazione del piano euclideo è sufficiente eseguire *iterativamente* le seguenti operazioni:

1. costruire una figura simile a quella della figura precedente di dimensioni doppie;

2. applicare la decomposizione di Conway a ciascuno dei cinque triangoli interni.

Il nostro scopo è ora di implementare questa costruzione in Dr. Geo utilizzando una Figura Scheme ricorsiva. La strategia che seguiamo è leggermente differente da quella descritta sopra, anche se equivalente. Costruiremo una Figura Scheme in modo che dato un triangolo rettangolo grande a piacere questo venga decomposto successivamente secondo Conway fino ad un certo livello di profondità. Un codice Scheme che realizza la costruzione è il seguente:

```
(new-figure "Pinwheel")

 (define (triangolo a b c k)

   (let* (
      (ab (Segment "" extremities a b))
      (bc (Segment "" extremities b c))
      (ca (Segment "" extremities c a))
      (m  (Point "" middle-2pts a b))
      (mn (Line "" orthogonal m ca))
      (n  (Point "" intersection mn ca))
      (bo (Line "" orthogonal b ca))
      (o  (Point "" intersection bo ca))
```

```
      (p  (Point "" middle-2pts b o)))

      ;; decomposizione di Conway

    (send m  masked)
    (send mn masked)
    (send n  masked)
    (send bo masked)
    (send o  masked)
    (send p  masked)

      (if (> k 0)

       (begin
          (triangolo a n m (- k 1 ))
          (triangolo o n m (- k 1 ))
          (triangolo m p o (- k 1 ))
          (triangolo m p b (- k 1 ))
          (triangolo b o c (- k 1 ))))
  )
)

(lets Point "A" free -6 0)
(lets Point "B" free  6 0)
(lets Point "C" free  6 6)
(triangolo A B C 3)
```

L'analisi delle parti significative del codice è abbastanza semplice una volta ricordato quanto appreso in precedenza. Nella parte iniziale viene descritta la decomposizione di Conway. Il ciclo permette di costruire le decomposizioni successive che, nello spirito, non differiscono di molto da quelle esaminate in precedenza, se non per il fatto importante che l'iterazione viene ora eseguita *simultaneamente su più figure*. La figura che si ottiene come risultato è la 3.5.

Risulta di interesse matematico notare come i triangoli che formano la tassellazione cambino orientamento ad ogni generazione. Più precisamente si può dimostrare che in una tassellazione di Conway del piano le orientazioni con cui i triangoli compaiono sono infinite. Ciò spiega anche il significato del nome della tassellazione. Se pensiamo ad un mulino a vento fatto con teli triangolari è facile comprendere infatti che quando il vento soffia le orientazioni con cui si presenta un singolo telo sono infinite!

Esercizio 13 *Scrivere il codice della figura Scheme che genera una tassellazione di Conway e la sua simmetrica rispetto a un cateto.*

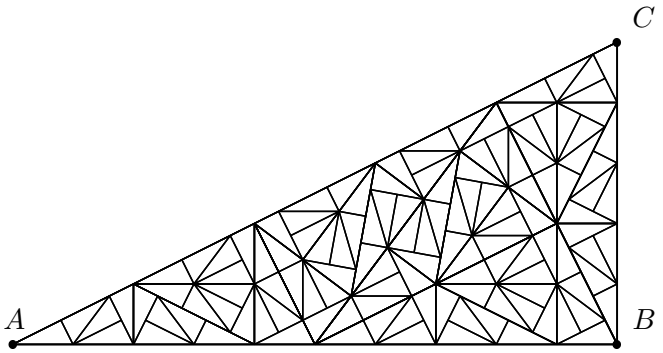


Figura 3.5: Tassellazione del mulino a vento

3.3.5 Insiemi autosimili

Il secondo tema che affrontiamo ricorrendo alle Figure Scheme per Dr. Geo riguarda gli **insiemi autosimili**. Gli insiemi autosimili hanno ricevuto in matematica una ragguardevole attenzione per le loro incredibili proprietà e per il loro legame con il concetto di *frattale*. Una definizione rigorosa di frattale esula dagli scopi di questo libro, tuttavia in modo intutivo possiamo dire che un insieme frattale è un oggetto la cui **dimensione** non può essere rappresentata da un numero intero.
Per cercare di addentrarci nel mondo degli insiemi autosimili cominciamo a studiare un esempio celebre: la *curva di Koch*. Si tratta di un esempio semplice e istruttivo. Iniziamo considerando un segmento $S_0 = AB$.

1. Prima iterazione: possiamo dividere S_0 in tre segmenti uguali, costruire un triangolo equilatero avente per base il segmento intermedio e rimuovere la base ottenendo la spezzata S_1 formata da quattro segmenti della stessa lunghezza rappresentata in figura 3.6.

2. Seconda iterazione: applichiamo a ciascuno dei quattro segmenti che formano S_1 la stessa procedura applicata in precedenza al segmento S_0.

Se si continua nello stesso modo si ottiene una successione infinita di spezzate il cui limite si chiama curva di Koch. Il codice Scheme che genera la figura 3.7 è il seguente:

```
(new-figure "Koch")

(define (vonkoch a b k)
```

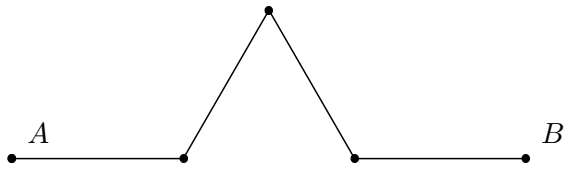


Figura 3.6: Costruzione della curva di Koch

```
  (if (> k 0)

    (let* (

                ;; costruzione geometrica

              (ab (Segment "" extremities a b))
              (c  (Point "" on-curve ab (/ 1 3)))
              (d  (Point "" on-curve ab (/ 2 3)))
              (m  (Point "" middle-2pts a b))
              (p  (Line "" orthogonal m ab))
              (ci (Circle "" 2points c d))
              (v  (Point "" intersection2 p ci))
          )

               ;; nascondiamo la costruzione

    (map (lambda (x) (send x masked)) (list ab c d m p ci v))

    ;; ricorsione

    (vonkoch a c (- k 1 ))
    (vonkoch c v (- k 1 ))
    (vonkoch v d (- k 1 ))
    (vonkoch d b (- k 1 ))

    )

    ;; costruzione di livello zero

    (Segment "" extremities a b)
   )
)
```

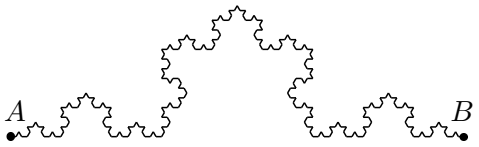


Figura 3.7: Curva di Koch

```
(lets Point "A" free -8 3)
(lets Point "B" free 8 3)

(vonkoch A B 5)
```

La struttura del codice è solo in parte simile a quelli visti in precedenza e quindi sono necessarie alcune osservazioni:

1. il nascondimento degli oggetti della costruzione è stato fatto utilizzando la funzione `map` la cui sintassi è `(map (lambda (oggetto) (send oggetto attributo) (list oggetti))`: l'attributo viene in questo modo replicato a tutti gli oggetti della lista senza dover ogni volta ripetere per ciascuno di essi `send masked oggetto`;

2. c'è un gioco sottile tra chiamata alla costruzione di livello zero e nascondimento degli oggetti nel corpo della funzione il cui effetto finale è di visualizzare *solo* l'ultimo output del ciclo tralasciando tutti i passaggi intermedi. Ciò può essere interessante in quanto la presenza delle costruzioni intermedie potrebbe rendere meno evidente la forma dell'oggetto geometrico limite.

Esercizio 14 *Modificare il codice precedente in modo da generare un* ***fiocco di neve*** *ossia un triangolo equilatero in cui ciascun lato è una curva di Koch di colore diverso. (Suggerimento: basta aggiungere una piccola parte di codice alla fine del codice che genera la curva di Koch.)*

Esercizio 15 *Modificare il codice che genera la curva di Koch in modo che gli estremi della curva siano due punti casuali.*

Una caratteristica della curva di Koch è stata definita da Benoit Mandelbrot [10] con il termine di *autosimiglianza*. Possiamo esprimere il concetto di autosimiglianza con un'immagine: supponiamo di avere a disposizione una lente di ingrandimento che ci permetta di ingrandire a piacere alcune parti della curva di Koch. Non è difficile convincersi del fatto che il panorama che vedremo di volta in volta sarà sempre identico a quello che vediamo

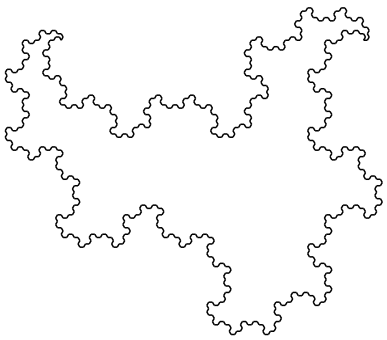


Figura 3.8: Pseudofiocco

guardando il tutto. In altri termini la curva è fatta di parti che sono simili al tutto. Si può esperire, almeno in parte, questo concetto deformando dinamicamente in Dr. Geo le figure ottenute.

3.3.6 Mostri

Lo studio degli insiemi autosimili è stato molto intenso nella prima metà del novecento ed ha portato alla scoperta di molti esempi interessanti [6]. Il primo esempio che analizziamo è lo **pseudofiocco di neve** di figura 3.8.

Il codice Scheme che genera questa figura non è direttamente disponibile ma viene lasciato al lettore per esercizio in quanto molto simile a quello utilizzato per disegnare la curva di Koch! In ogni caso trattandosi di un esercizio non del tutto banale diamo alcuni suggerimenti:

1. la figura, analogamente al fiocco di neve, è fatta di tre parti identiche e disposte secondo i lati di un triangolo equilatero;

2. una singola parte si ottiene ricorsivamente a partire da un segmento S_0: la prima iterazione si ottiene costruendo un triangolo isoscele di base S_0, con angoli alla base di ampiezza 30° e considerando come oggetto S_1 la spezzata formata dai due lati obliqui del triangolo; la seconda iterazione consiste nel ripetere opportunamente la costruzione precedente su ciascuno dei due lati obliqui che formano S_1.

Un secondo esempio curioso è il **drago di Sierpinski**. Il codice della Figura Scheme che permette di ottenerlo è il seguente:

```
(new-figure "Sierpinski Dragon")
```

```
(define (dragon a b k)

    (if (> k 0)
(let* (

(c1 (Circle "" 2points a b ))
(c2 (Circle "" 2points b a ))
(c  (Point "" intersection2 c1 c2))
(m  (Point "" middle-2pts a c))
(n  (Point "" middle-2pts b c))
(am (Segment "" extremities a m))
(mn (Segment "" extremities m n))
(nb (Segment "" extremities n b))
       )

  ;; nascondiamo la costruzione

  (map (lambda (x) (send x masked))
       (list c1 c2 c m n am mn nb))

  ;; ricorsione

  (dragon m a (- k 1 ))
  (dragon m n (- k 1 ))
          (dragon b n (- k 1 ))
  )

;; livello zero: traccia un segmento

(Segment "" extremities a b)
)
    )



(lets Point "A" free -4 1.5)
(lets Point "B" free 4 1.5)

(dragon A B 5)
```

La figura generata è la 3.9.

Esercizio 16 *Dopo aver studiato il codice Scheme, spiega quale procedimento geometrico iterativo permette di generare il drago di Sierpinski.*

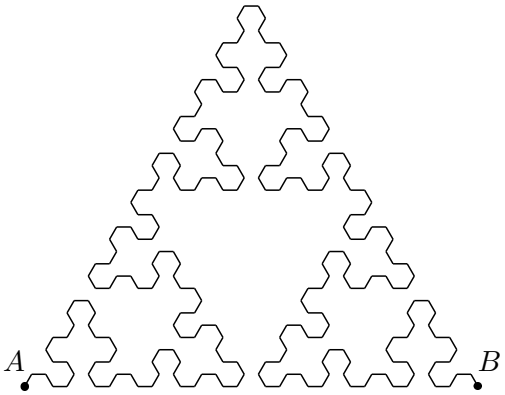


Figura 3.9: Drago di Sierpinski

Capitolo 4

Trasformazioni del Piano

Prima di affrontare la costruzione di altre figure Scheme è importante impadronirsi del linguaggio delle **trasformazioni del piano**. Lo studio delle trasformazioni del piano riveste un'importanza centrale per un approccio moderno allo studio della Geometria.

4.1 Riflessione

Consideriamo nel piano una retta r e un punto P come in figura 4.1. Utilizzando Dr. Geo possiamo eseguire la seguente costruzione:

- traccia la retta s perpendicolare per P alla retta r;
- detto H il punto di intersezione tra le rette s e r disegna la circonferenza $\mathcal{C}$ di centro H e passante per P;
- la circonferenza $\mathcal{C}$ interseca la retta s in P e P'

il punto P' si dice punto **riflesso** di P rispetto alla retta r. La retta r prende il nome di **asse di riflessione**. Osserviamo che, per costruzione, $HP = HP'$ e che quindi H è punto medio del segmento PP'.
La costruzione descritta sopra funziona per un qualsiasi punto P non appartenente all'asse di riflessione. Nel caso di un qualsiasi punto appartenente all'asse di riflessione poniamo **per definizione** che esso coincida con il suo punto riflesso. Nel seguito chiameremo punti **uniti** di una traformazione i punti che coincidono con i loro trasformati. Nel caso della riflessione allora i punti appartenenti all'asse di riflessione sono punti uniti.

4.1.1 Riflessione di oggetti geometrici

Dopo aver imparato a riflettere i punti ci poniamo il problema di riflettere altri oggetti geometrici. Iniziamo da rette, semirette e segmenti osservando che ciascuno di questi oggetti è definito assegnando una coppia distinta di

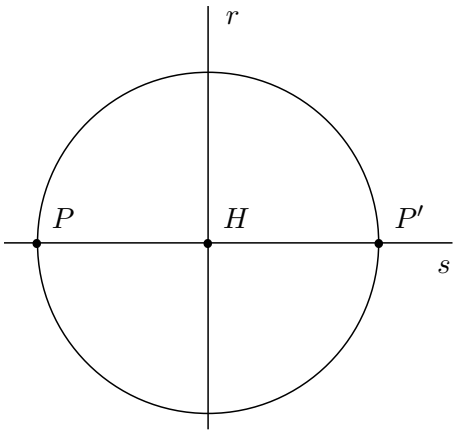


Figura 4.1: Riflessione

punti del piano. Per comprendere come possiamo costruire il loro riflesso rispetto ad una retta r è sufficiente studiare ad esempio il caso di un segmento AB. Per riflettere AB determiniamo in primo luogo i punti A' e B' che sono i riflessi di A e B rispetto ad r.

Il riflesso di AB rispetto ad r è il segmento $A'B'$.

In figura 4.2 sono riprodotti i diversi casi che si possono presentare nella riflessione di un segmento. Osserviamo che nella riflessione di un segmento possono darsi tre situazioni: non ci sono **punti uniti**, c'è un punto unito, ci sono infiniti punti uniti.
Il riflesso di un poligono è il poligono che ha per lati i riflessi dei lati del poligono di partenza, mentre il riflesso di una circonferenza è la circonferenza che ha per centro il riflesso del centro della circonferenza di partenza e raggio uguale al raggio di quest'ultima. Ricordiamo che spesso il termine riflessione viene sostituito con l'equivalente di *simmetria assiale*.

Esercizio 17 *Utilizzando in Dr. Geo la trasformazione di riflessione costruisci a partire da un triangolo rettangolo T il suo riflesso T' rispetto a un cateto e T'' rispetto all'ipotenusa. Sai dire che figure geometriche sono le figure formate rispettivamente da T e T' e da T e T''?*

4.2 Rotazione

La seconda trasformazione di cui ci occupiamo è la **rotazione**. Consideriamo innanzitutto due rette r e s che formano tra loro un certo angolo, per fissare un'idea, di 60°. Dato un triangolo ABC eseguiamo su esso due riflessioni come in figura 4.3. Se chiamiamo il triangolo finale $A'B'C'$ esso è ottenuto ruotando ABC intorno al punto K (centro di rotazione) di un

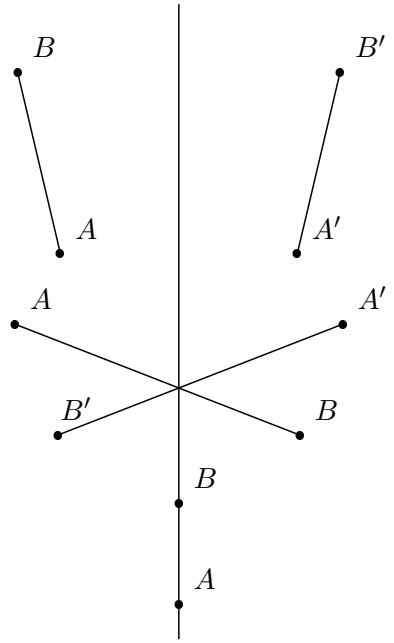


Figura 4.2: Riflessione di un segmento

angolo pari a 120°. In effetti se dopo aver riprodotto la figura con Dr. Geo si misurano gli angoli $\angle C'KC$, $\angle B'KB$ e $\angle A'KA$ si osserverà che il loro valore è sempre di 120°.

Possiamo allora dare la seguente definizione:

Definizione 1 *Dato un oggetto geometrico una rotazione di centro K e di un angolo 2α è la trasformazione ottenuta eseguendo successivamente due riflessioni dell'oggetto intorno ad una coppia di rette incidenti in K e formanti tra loro un angolo di ampiezza α.*

4.2.1 Angolo radiante

In Dr. Geo disegna due semirette OA e OB di origine comune O. Sulla semiretta OB fissa un punto mobile M e disegna la circonferenza di centro O e passante per M. Con il goniometro di Dr. Geo misura l'ampiezza α_g dell'angolo $\angle AOM$. Il risultato della misura è espresso in gradi. Tuttavia in Matematica è preferibile misurare gli angoli utilizzando i **radianti**. Per comprendere il concetto di radiante utilizzando Dr. Geo procediamo come segue:

- sull'arco AB fissa un punto C qualsiasi;
- costruisci l'arco passante per i punti A, B e C;

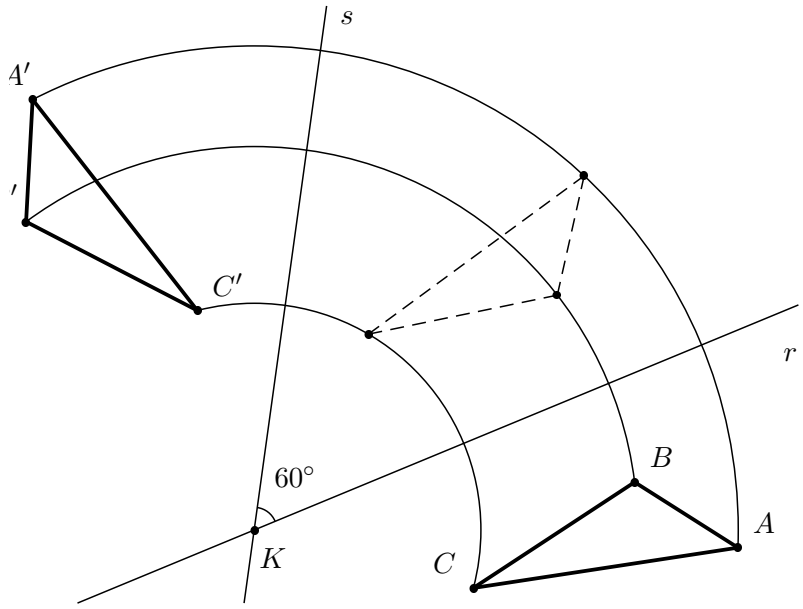


Figura 4.3: Rotazione di un triangolo

- nascondi la circonferenza di centro O e il punto C;
- scrivi uno script Guile che calcola il rapporto α tra la lunghezza dell'arco e il raggio della circonferenza cui appartiene l'arco (basta prendere la lunghezza di OM)

Non è difficile verificare (farlo!) che vale la seguente proporzione:

$$\frac{\alpha}{\alpha_g} = \frac{2\pi}{360}.$$

Se adesso deformiamo dinamicamente la figura spostando il punto M si noterà che l'uscita dello script α non cambia. Il numero α definisce l'ampiezza in radianti dell'angolo $\angle AOM$. La proporzione precedente permette di calcolare a partire dall'angolo in gradi il valore dell'angolo in radianti e viceversa.

Esercizio 18 *Determinare l'ampiezza in radianti degli angoli di* $30°$, $60°$, $45°$ *e* $90°$.

4.3 Traslazione

La terza trasformazione del piano di cui ci occupiamo è la **traslazione**. Il procedimento che seguiamo per definire la traslazione è analogo a quello già

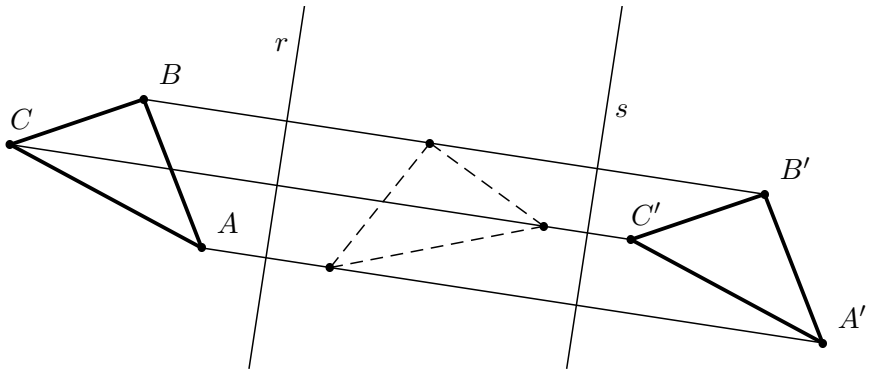


Figura 4.4: Traslazione di un triangolo

visto per la rotazione. Consideriamo una coppia di rette r e s parallele. Dato un triangolo ABC eseguiamo una sua riflessione intorno ad r e successivamente una riflessione intorno ad s in modo da ottenere il triangolo $A'B'C'$. In geometria diciamo che il triangolo $A'B'C'$ è una traslazione di ABC. Osserviamo che la distanza tra una qualsiasi coppia di vertici corrispondenti di ABC e $A'B'C'$ è pari a $2d$, dove d indica la distanza tra le due rette parallele. Per descrivere la traslazione si ricorre al **vettore di traslazione** che si rappresenta con un qualsiasi segmento di lunghezza $2d$, perpendicolare ad r e diretto (punta del vettore) da ABC verso $A'B'C'$. In generale:

Definizione 2 *Dato un oggetto geometrico una traslazione è la trasformazione che si ottiene eseguendo in successione due riflessioni dell'oggetto intorno a due rette r e s parallele.*

Riflessioni, rotazioni e traslazioni sono tre esempi trasformazioni **isometriche** del piano. Il termine isometrico significa che queste trasformazioni *lasciano invariate le distanze*. Tra le trasformazioni isometriche dovremmo includere anche la **glissoriflessione** il cui studio viene lasciato come approfondimento [5] al lettore.

Esercizio 19 *In Dr. Geo è implementata anche la trasformazione di **simmetria centrale**. Utilizzando quanto sai sulle isometrie descrivi questa trasformazione.*

4.4 Omotetie

Molte trasformazioni del piano non sono isometriche. Nell'insieme delle trasformazioni non isometriche spiccano le trasformazioni **omotetiche**. Per

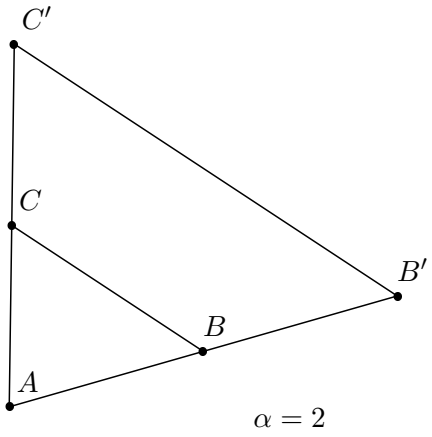


Figura 4.5: Omotetia di un triangolo

comprendere di che tipo di trasformazioni si tratta analizziamo come esempio la trasformazione omotetica di un fattore $\alpha = 2$ e di centro A di un triangolo ABC.

Il risultato è un triangolo $AB'C'$ che ha tutti i lati di lunghezza doppia rispetto a quello di partenza e notiamo che i lati AB' e AC' giacciono sulle rette AB e AC mentre $B'C'$ e BC sono paralleli. Notiamo ancora che il vertice A (centro dell'omotetia) è un punto unito della trasformazione. Non è difficile immaginare che se $\alpha = 1/2$ si sarebbe ottenuto un triangolo simile ad ABC ma con i lati di lunghezza dimezzata.

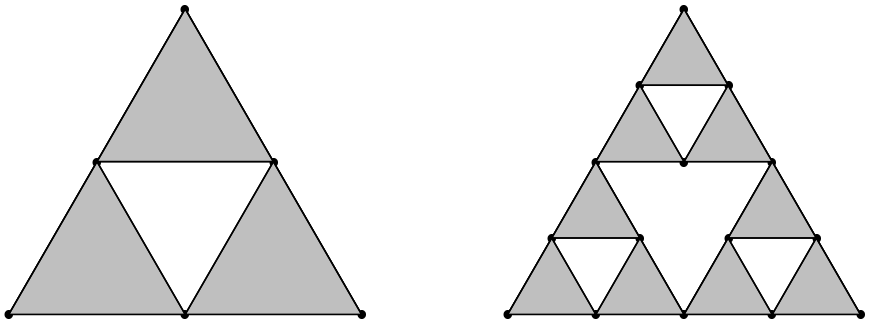
Esercizio 20 *Fissato un centro di omotetia C e un fattore α trasforma con Dr. Geo delle circonferenze a tua scelta. Verifica in ogni caso se le trasformazioni sono conformi alle tue aspettative.*

4.5 Poligoni di Sierpinski

In questo paragrafo analizziamo una classe di insiemi autosimili interessanti di cui alcuni spiccano per bellezza estetica.

4.5.1 Triangolo di Sierpinski

L'esempio semplice del **triangolo di Sierpinski**, che trattiamo in modo dettagliato, ci permette da un lato di capire alcuni motivi che lo rendono matematicamente interessante e dall'altro come scrivere una Figura Scheme che permetta di disegnarlo.



Prima Iterazione Seconda Iterazione

Figura 4.6: Costruzione del triangolo di Sierpinski

Iniziamo allora considerando il triangolo equilatero S_0 di figura e supponiamo che il suo lato sia di lunghezza unitaria.

1. Prima Iterazione: possiamo dividere S_0 in quattro triangoli equilateri di lato $1/2$ congiungendo i punti medi dei suoi lati. Il triangolo nel centro è ruotato rispetto ai rimanenti tre di $180°$. Se ora rimuoviamo i punti interni del triangolo centrale, non i suoi lati, otteniamo un nuovo insieme piano, contenuto in S_0, che chiamiamo S_1.

2. Seconda Iterazione: applichiamo ai tre triangoli equilateri di lato $1/2$ che formano S_1 la stessa procedura applicata in precedenza al triangolo S_0. In questo modo otterremo l'insieme S_2 rappresentato sempre in figura.

Se si continua nello stesso modo si ottiene una successione infinita di insiemi piani il cui limite S è il triangolo di Sierpinski. Dal momento che la successione di insiemi piani che si ottiene è decrescente, nel senso che ogni insieme è contenuto nel successivo, il suo limite sarà contenuto nel triangolo di partenza.
Analizziamo ora in dettaglio alcune proprietà di S:

Teorema 3 *Il perimetro di S è infinito.*

Dimostrazione Dal momento che S_0 ha lato unitario il suo perimetro è 3. Se calcoliamo il perimetro di S_1 avremo:

$$p_1 = 3 \cdot \frac{3}{2} = \frac{9}{2} > \frac{3}{2}$$

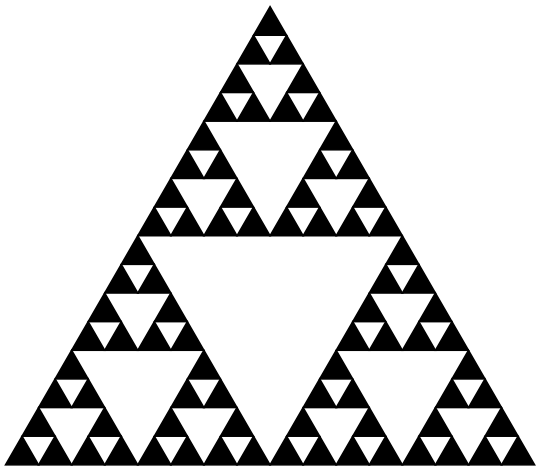


Figura 4.7: Triangolo di Sierpinski

che è pari al perimetro di 3 triangoli di lato 1/2. Il perimetro di S_2 sarà

$$p_2 = 9 \cdot \frac{3}{4} = \frac{27}{4} > \left(\frac{3}{2}\right)^2.$$

Dopo n iterazioni avremo:

$$p_n > \left(\frac{3}{2}\right)^n$$

da cui si comprende che al tendere di n all'infinito il perimetro di S è sicuramente infinito. □

Teorema 4 *L'area di S è nulla.*

La dimostrazione di questo teorema viene lasciata per esercizio in quanto concettualmente del tutto simile a quella del teorema precedente.

Sottolineiamo invece che la situazione, da un punto di vista squisitamente matematico, è curiosa in quanto i concetti di area e di perimetro risultano del tutto inefficaci per descrivere le caratteristiche geometriche del triangolo di Sierpinski. Questo lascia presupporre che S, detto in modo molto semplice, sia qualche cosa di più di un oggetto unidimensionale (una linea) e qualche cosa di meno di un oggetto bidimensionale (una superficie). L'approfondimento di questi aspetti esula dagli scopi di questo libro e viene rimandatata alla bibliografia [6].
Vediamo ora come implementare il triangolo di Sierpinsli come Figura Scheme ricorsiva.

```
(new-figure "Sierpinski")

(define (triangolo a b c k)

  (if (> k 0)

      (let* (
     ;; lati del triangolo

     (ab (Segment "" extremities a b))
     (bc (Segment "" extremities b c))
     (ca (Segment "" extremities c a))


     ;; vertici nuovi triangoli

     (n (Point "" middle-2pts  a b ))
     (m (Point "" middle-2pts  b c ))
     (p (Point "" middle-2pts  c a ))

     ;; triangoli

     (p1 (Polygon "" npoints a n p))
     (p2 (Polygon "" npoints n b m))
     (p3 (Polygon "" npoints p m c)))

;; nascondiamo la costruzione

    (map (lambda (x) (send x masked))
       (list ab bc ca n m p p1 p2 p3))
    ;; ricorsione

    (triangolo a n p (- k 1 ))
    (triangolo n b m (- k 1 ))
    (triangolo p m c (- k 1 )))

      ;; livello zero

      (Polygon "" npoints a b c)
      )
  )

;; costruzione triangolo equilatero
```

```
(lets Point "A" free 0 0)
(lets Point "B" free 7 0)
(lets Circle "c" 2points B A)
(lets Circle "k" 2points A B)
(lets Point "C" intersection c k)

(send c masked)
(send k masked)
(send A masked)
(send B masked)
(send C masked)

(triangolo A B C 4)
```

Lo studio del codice è lasciato al lettore e si suggerisce di condurlo confrontandolo con il codice utilizzato per realizzare la curva di Koch.

Esercizio 21 *Scrivere una Figura Scheme che permetta di ottenere una costruzione analoga al triangolo di Sierpinski partendo da un quadrato. L'idea è di dividere il quadrato di partenza in nove quadrati congruenti, di rimuovere il quadrato centrale e quindi di iterare la costruzione sui quadrati rimanenti. La figura che si ottiene è nota come* ***tappeto di Mazurkiewicz*** *ed ha interessanti proprietà topologiche.*

Esercizio 22 *Costruire una Figura Scheme che realizzi un esagono di Sierpinski.*

4.5.2 Pentagono di Dürer

Costruire un pentagono di Sierpinski è più complesso dei casi fino ad ora trattati in quanto non è possibile suddividere un pentagono regolare con pentagoni regolari senza lasciare spazi vuoti.
Prima di analizzare la questione e visto che Dr. Geo non comprende tra le sue costruzioni base quella del pentagono regolare affrontiamo questo punto. Ci sono diverse costruzioni con riga e compasso che conducono alla costruzione di pentagono regolare. Quella che scegliamo è contenuta in [14] e permette di costruire un pentagono regolare inscritto in una circonferenza di raggio dato:

- consideriamo la circonferenza $\mathcal{C}$ di centro O e passante per il punto A che come vedremo sarà un vertice del pentagono;
- tracciamo la retta perpendicolare ad AO per il punto O e indichiamo con B e C i suoi punti di intersezione con $\mathcal{C}$;
- costruiamo il punto medio M del raggio OC;

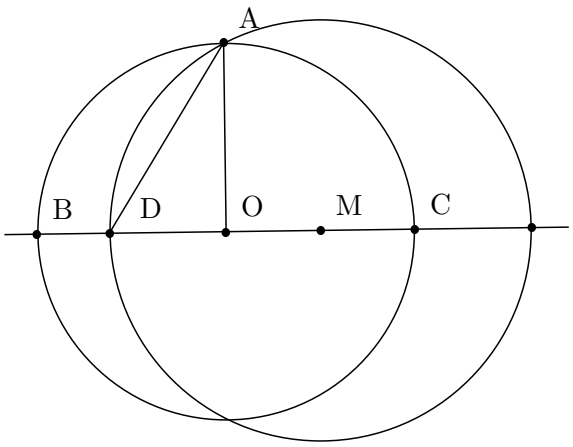


Figura 4.8: Costruzione tolemaica

- tracciata la circonferenza di centro M e passante per A essa interseca il raggio BO in D;
- Tolomeo dimostra che il segmento AD ha lunghezza pari al lato di un pentagono regolare inscritto in $\mathcal{C}$.

Non è difficile completare la costruzione del pentagono regolare riportando AD sulla circonferenza $\mathcal{C}$ e costruendo i rimanenti quattro lati. Possiamo trascrivere questa costruzione come Figura Scheme:

```
(new-figure "Pentagono")

(lets Point "O" free 0 0) ;; centro circonferenza
(lets Point "A" free 0 3) ;; vertice del pentagono

;; costruzione tolemaica di un pentagono

(lets Line "r" 2points O A)
(lets Circle "c" 2points O A)
(lets Line "p" orthogonal O r)
(lets Point "T" intersection p c)
(lets Point "M" middle-2pts O T)
(lets Circle "k" 2points M A)
(lets Point "N" intersection2 k p)
(lets Circle "h" 2points A N)
(lets Point "B" intersection h c)
(lets Point "E" intersection2 h c)
```

```
(lets Angle "alfa" geometric E A B)
(lets Point "D" rotation A E alfa)
(lets Point "C" reflexion D r)

;; nascondiamo alcuni oggetti

(map (lambda (x) (send x masked))
(list r p T M k N h alfa))

;; costruzione dei lati del pentagono

(lets Segment "" extremities A B)
(lets Segment "" extremities B C)
(lets Segment "" extremities C D)
(lets Segment "" extremities D E)
(lets Segment "" extremities E A)
```

che il lettore può valutare con Dr. Geo.
Passiamo ora al problema della decomposizione di un pentagono in pentagoni regolari. A questo scopo possiamo approfittare di quanto contenuto nel *Manuale del Pittore* di Albrecht Dürer del 1525 [9]. Un capitolo del libro è dedicato alle tassellazioni e tra esse viene menzionata una tassellazione pentagonale che suggerisce la decomposizione di figura 4.9.

La decomposizione si realizza in due momenti. In una prima fase (vedi figura 4.9) si costruisce, a partire da un pentagono regolare, una stella pentagonale la cui parte centrale individua un pentagono regolare ruotato rispetto a quello di partenza di 180° e riscalato di un fattore

$$\alpha = \frac{3 - \sqrt{5}}{2}.$$

Questa prima costruzione ha un'origine molto antica [1] e secondo la tradizione ha permesso ai Pitagorici di scoprire il numero irrazionale

$$\tau = \frac{1 + \sqrt{5}}{2}$$

noto in letteratura come **numero aureo**. Si tratta di un numero molto importante per la sua comparsa nelle più diverse aree del sapere: biologia, musica, arte, fisica e via di seguito. Il lettore interessato ad approfondire questo aspetto affascinante è rinviato all'ottimo [9].
In una seconda fase si costruiscono altri cinque pentagoni congruenti al pentagono centrale e disposti come in figura 4.9. Come si può notare il pentagono di partenza non viene completamente ricoperto dai sei pentagoni riscalati ma rimangono delle aree vuote che sono state evidenziate in nero.

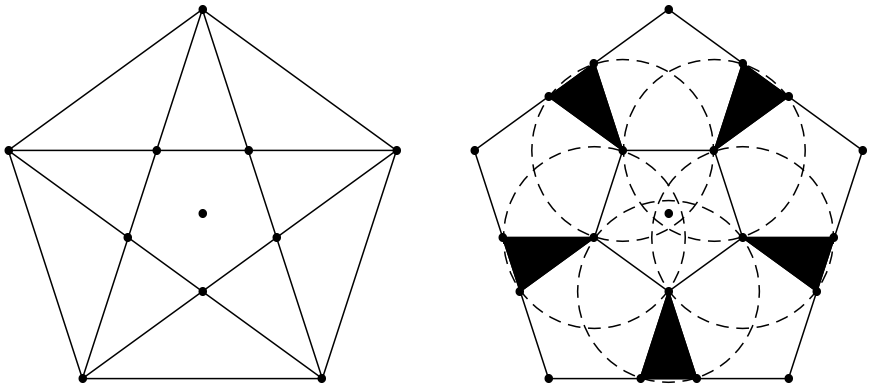


Figura 4.9: Costruzione di Dürer

Nello spirito della costruzione del triangolo di Sierpinski possiamo iterare la costruzione appena vista su ciascuno dei sei pentagoni riscalati. Per fare questo utilizziamo la seguente Figura Scheme:

```
 (new-figure "Durer")

(define (durer a b c d e k)

  (if (> k 0)

      (let* (
     ;; lati del pentagono

     (ab (Segment "" extremities a b))
     (bc (Segment "" extremities b c))
     (cd (Segment "" extremities c d))
     (de (Segment "" extremities d e))
     (ea (Segment "" extremities e a))

     ;; pentagramma

     (ac (Segment "" extremities a c))
     (ad (Segment "" extremities a d))
     (bd (Segment "" extremities b d))
     (be (Segment "" extremities b e))
     (ce (Segment "" extremities c e))

     ;; pentagono centrale
```

```
     (f (Point "" intersection ac be))
     (g (Point "" intersection ac bd))
     (h (Point "" intersection ce bd))
     (l (Point "" intersection ce ad))
     (m (Point "" intersection ad be))

     ;; circonferenze ausiliarie

     (c1 (Circle "" 2points h g))
     (c2 (Circle "" 2points l h))
     (c3 (Circle "" 2points m l))
     (c4 (Circle "" 2points f m))
     (c5 (Circle "" 2points g f))

     ;; vertici nuovi pentagoni

     (n (Point "" intersection  cd c1))
     (o (Point "" intersection2 cd c1))
     (p (Point "" intersection  c2 de))
     (q (Point "" intersection2 c2 de))
     (r (Point "" intersection  c3 ea))
     (s (Point "" intersection2 c3 ea))
     (t (Point "" intersection  c4 ab))
     (u (Point "" intersection2 c4 ab))
     (v (Point "" intersection  c5 bc))
     (z (Point "" intersection2 c5 bc))

     ;; poligoni

     (p1 (Polygon "" npoints c n h g z))
     (p2 (Polygon "" npoints o d p l h))
     (p3 (Polygon "" npoints l q e r m))
     (p4 (Polygon "" npoints m s a t f))
     (p5 (Polygon "" npoints v g f u b))
     (p6 (Polygon "" npoints f g h l m)))

;; nascondiamo la costruzione

    (map (lambda (x) (send x masked))
       (list c1 c2 c3 c4 c5
     n o p q r s t u v z f g h l m
     ab bc cd de ea ac ad bd be ce
     p1 p2 p3 p4 p5 p6))
```

```
    ;; ricorsione

    (durer g z c n h (- k 1 ))
    (durer l h o d p (- k 1 ))
    (durer r m l q e (- k 1 ))
    (durer a t f m s (- k 1 ))
    (durer u b v g f (- k 1 ))
    (durer f g h l m (- k 1 )))

      ;; livello zero

      (Polygon "" npoints a b c d e)
      )
  )

;; costruzione tolemaica di un pentagono

(lets Point "O" free 0 0) ;; centro circonferenza

(lets Point "A" free 0 3.5) ;; vertice del pentagono

(lets Line "r" 2points O A)
(lets Circle "c" 2points O A)
(lets Line "p" orthogonal O r)
(lets Point "T" intersection p c)
(lets Point "M" middle-2pts O T)
(lets Circle "k" 2points M A)
(lets Point "N" intersection2 k p)
(lets Circle "h" 2points A N)
(lets Point "B" intersection h c)  ;; vertice del pentagono
(lets Point "E" intersection2 h c) ;; vertice del pentagono
(lets Angle "alfa" geometric E A B)
(lets Point "D" rotation A E alfa) ;; vertice del pentagono
(lets Point "C" reflexion D r)     ;; vertice del pentagono


(map (lambda (x) (send x masked))
       (list r c p T M k N h alfa O A B C D E))

(durer A B C D E 4)
```

Valutando la Figura Scheme si ottiene la figura 4.10 nota come **pentagono di Dürer**.

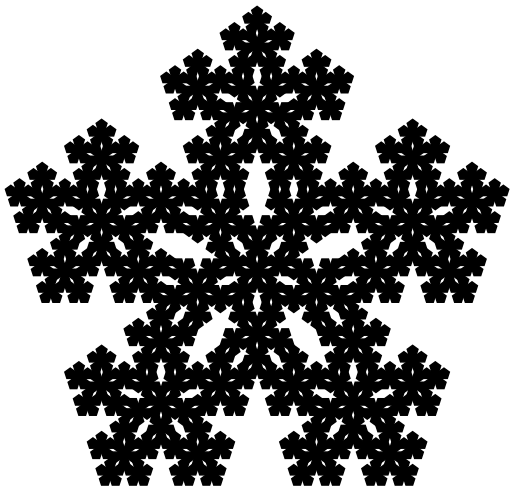


Figura 4.10: Pentagono di Dürer

Se si vuole essere pignoli, dobbiamo ammettere che nonostante la costruzione del pentagono di Dürer abbia in comune con la costruzione del triangolo di Sierpinski sia l'aspetto iterativo che l'idea di decomporre un poligno regolare in poligoni ad esso simili, vi è un aspetto in cui essa si diversifica. Nella costruzione di Sierpinski, una volta effettuata la decomposizione del poligono, il poligono centrale viene rimosso e il processo iterativo avviene solo sui rimanenti poligoni.

Esercizio 23 *Modifica il codice che produce il pentagono di Dürer rimuovendo il pentagono centrale in modo da ottenere un pentagono di Sierpinski.*

4.6 Botanica

In questo paragrafo vediamo come costruire delle Figure Scheme che hanno una certa somiglianza con piante e arbusti. Quanto contenuto nel seguito è ispirato o tratto da [12].
In primo luogo prendiamo le mosse dalla costruzione geometrica *base* di figura 4.11. L'oggetto geometrico di partenza è il segmento AB e a partire da esso viene eseguita la costruzione di quelli che chiamiamo i rami di *primo livello* (EM, BL e BR):

1. costruisci il punto medio M del segmento AB;

2. costruisci il vettore $\vec{v} = \overrightarrow{BM}$;

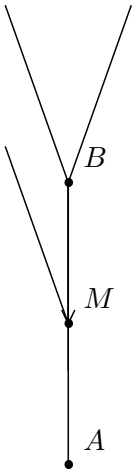


Figura 4.11: Costruzione base

3. costruisci il segmento s ottenuto applicando ad AB un'omotetia di centro B e di fattore 2/3;

4. il primo ramo è la rotazione di s intorno a B di un angolo di -2.8 radianti;

5. il secondo ramo è la rotazione di s intorno a B di un angolo di 2.8 radianti;

6. il terzo ramo è la traslazione del primo ramo di un vettore $\vec{v}$.

La costruzione potrebbe procedere applicando a ciascun ramo di primo livello la costruzione base e via di seguito. Tuttavia introduciamo una leggera complicazione (figura 4.12). Sui rami di primo livello costruiamo dei rami secondari a cui applicare la costruzione base:

1. primo ramo secondario BC è la rotazione di BM intorno a B di un angolo di -2.8 radianti;

2. il secondo ramo secondario è la rotazione di BM intorno a B di un angolo di 2.8 radianti;

3. il terzo ramo è la traslazione del primo ramo secondario di un vettore $\vec{v}$.

Per vedere la figura che si ottiene iterando la costruzione precedente scriviamo una Figura Scheme che implementa la costruzione:

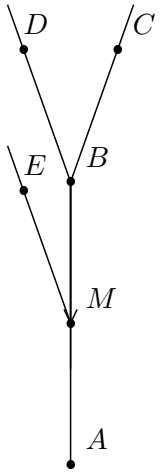


Figura 4.12: Rami secondari

```
(new-figure "Arbusto")

(define (tree a b k)
(let* (

       ;; costruzione dell'arbusto

       (ab (Segment "" extremities a b))
           (fac (Numeric "" free 3 3 (/ 2 3)))
           (m (Point "" middle-2pts a b))
           (v (Vector "" 2points b m))

       (bm (Segment "" scale ab b fac))

       (angl (Numeric "" free -3  3 2.8))
           (angr (Numeric "" free -3 -3 -2.8))

       (lbc (Segment "" rotation bm b angr))
       (rbc (Segment "" rotation bm b angl))
       (mbc (Segment "" translation lbc v))
       (lb (Point "" rotation m b angr))
       (rb (Point "" rotation m b angl))
       (mb (Point "" translation rb v)))
```

```
  ;; nascondiamo alcuni oggetti

  (map (lambda (x) (send x masked))
       (list fac m v bm angl angr lb rb mb))

  (if (> k 0)

  (begin
  (tree m mb (- k 1 ))
  (tree b lb (- k 1 ))
  (tree b rb (- k 1 ))))
       )
)

(lets Point "A" free 0 0)
(lets Point "B" free 0 6)

(send B masked)

(tree A B 6)
```

Il sorprendente risultato è rappresentato in figura 4.13!

Esercizio 24 *Scrivi una Figura Scheme che abbia per risultato una pianta. Hai diverse scelte: puoi costruirne una nuova, modificare la precedente aumentando il numero di rami di livello 1, cambiando l'inclinazione di alcuni rami, ...*

Un secondo esempio di pianta che proponiamo è generato dal seguente codice Scheme il cui studio viene lasciato come esercizio al lettore.

```
(new-figure "Albero")

(define pi (acos -1)) ;;pigreco

(define (branch a b k)

  (let* (
     (ab (Segment "" extremities a b))
     (m (Point "" middle-2pts a b))
     (am (Segment "" extremities a m))
     (mb (Segment "" extremities m b))

     ;; ramo destro
```

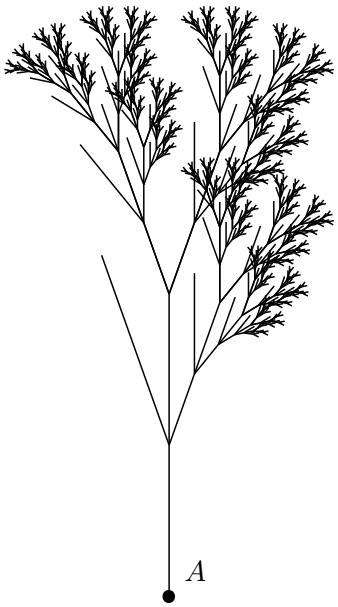


Figura 4.13: Arbusto

```
(alfa (Numeric "" free 5 5 (* (/ 135 180) pi)))
(c (Point "" rotation m b alfa))
(beta (Numeric "" free 6 6 (* (/ 202.5 180) pi)))
(d (Point "" rotation b c beta))
(e (Point "" rotation c d beta))
(bc (Segment "" extremities b c))
(cd (Segment "" extremities c d))
(de (Segment "" extremities d e))

 ;; ramo sinistro

(gamma (Numeric "" free 7 7 (* (/ 68.5 180) pi)))
(f (Point "" rotation c b gamma))
(delta (Numeric "" free 8 8 (* (/ 157.5 180) pi)))
(g (Point "" rotation b f delta))
(h (Point "" rotation f g delta))
(bf (Segment "" extremities b f))
(fg (Segment "" extremities f g))
(gh (Segment "" extremities g h))
```

```
     )

    ;; nascondiamo la costruzione

    (map (lambda (x) (send x masked))
 (list ab m alfa beta gamma delta c d e f g h ))

    (if (> k 0)

    (begin
      (branch a m (- k 1 ))
      (branch m b (- k 1 ))
      (branch b c (- k 1 ))
      (branch c d (- k 1 ))
      (branch d e (- k 1 ))
      (branch b f (- k 1 ))
      (branch f g (- k 1 ))
      (branch g h (- k 1 ))))))

(lets Point "A" free 0 -2)
(lets Point "B" free 0 1)
(lets Segment "s" extremities A B)

(send A masked)
(send B masked)

(send s thickness large)
(send s color black)

(branch A B 3)
```

La figura che si ottiene è 4.14.

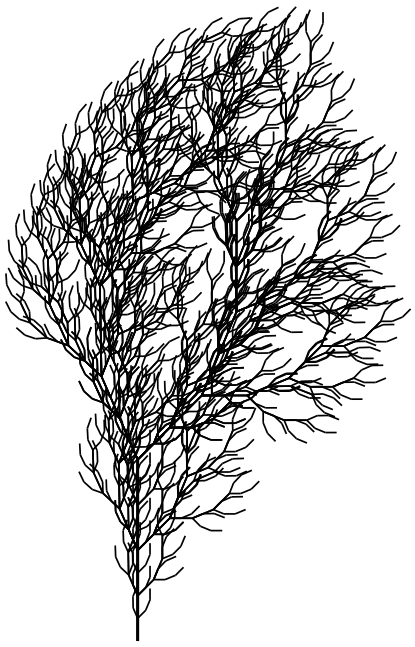


Figura 4.14: Albero

4.7 Insiemi di Fathauer

R. Fathauer ha realizzato diverse figure geometriche che coniugano in modo interessante arte e matematica. Le figure sono tutte costruite, a meno di piccole variazioni sul tema, seguendo un unico metodo. In questo paragrafo analizziamo in dettaglio una di queste costruzioni in modo da fornire gli strumenti per poterne riprodurre altre.

Il primo passaggio per costruire un insieme di Fathauer consiste nel realizzare un tassello base. Nel nostro caso si tratta dell'esagono $AGCFME$ costruito a partire da un triangolo equilatero ABC. Non è difficile riprodurre la figura 4.15 con Dr. Geo notando che: M è il punto medio del lato AC, G il baricentro di ABC, EG è parallelo ad AB, EA è perpendicolare ad AB.

Il secondo passaggio consiste nel realizzare la costruzione di primo livello: tipicamente si tratta di utilizzare dei tasselli simili a quello base e disporli in modo opportuno.

Quindi si tratta di iterare la costruzione. Il procedimento complessivo si può comprendere studiando la seguente Figura Scheme:

```
(new-figure "Fathtriangolo")

(define (tile a b k)
```

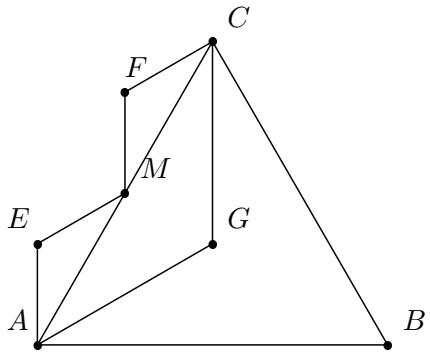


Figura 4.15: Tassello fondamentale

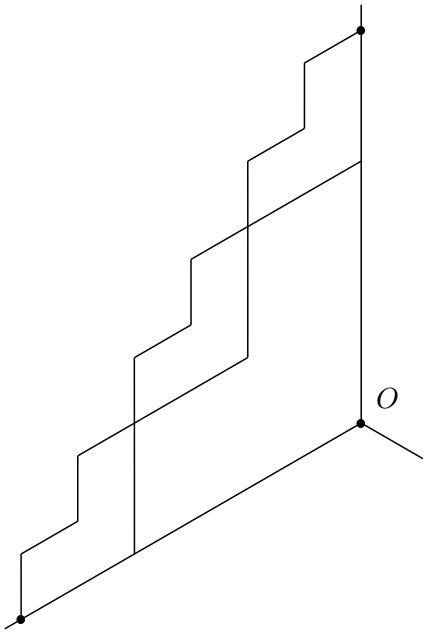


Figura 4.16: Costruzione di primo livello

```
  (let* (
 ;; costruzione geometrica

 (ab (Segment "" extremities a b))
 (alfa (Numeric "" free 0 11 -1.047))
 (s  (Point "" rotation a b alfa))
 (sb (Line "" 2points s b))

 (c (Point "" reflexion a sb))
 (bc (Segment "" extremities b c))
 (apar (Line "" parallel a bc))
 (bperp (Line  "" orthogonal b bc))
 (f  (Point "" intersection apar bperp))
 (d  (Point "" reflexion f sb))
 (e (Point "" middle-2pts a c))

 (cd (Segment "" extremities c d))
 (de (Segment "" extremities d e))
 (ef (Segment "" extremities e f))
 (fa (Segment "" extremities f a))

         (beta (Numeric "" free 0 15 2.094))
         (g  (Point "" rotation f a beta)))

    ;; nascondiamo parte della costruzione

    (map (lambda (x) (send x masked))
       (list ab alfa s sb c bc apar bperp f d e beta g))


    ;; ricorsione

    (if (> k 0)

(begin
  (tile g a (- k 1 ))
  (tile f e (- k 1 ))
  (tile d c (- k 1 ))))))

;; triangolo iniziale

(lets Point "A" free -3 0)
(lets Point "O" free  0 (sqrt 3))
(lets Point "B" free  3 0)
```

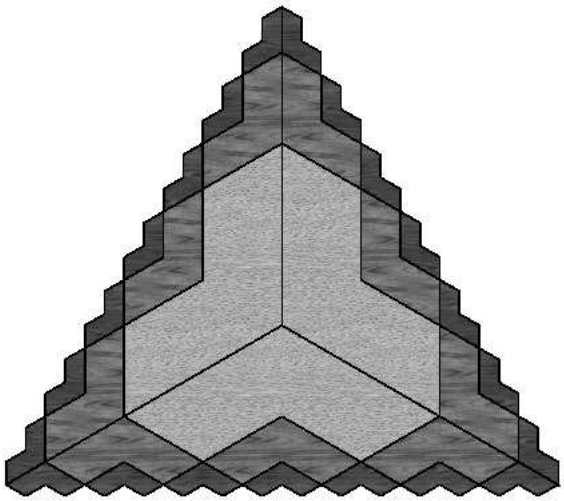


Figura 4.17: Insieme di Fathauer

```
(lets Point "C" free  0 (* 3 (sqrt 3)))
(lets Ray "" 2points O A)
(lets Ray "" 2points O B)
(lets Ray "" 2points O C)

(send A masked)
(send B masked)
(send C masked)
(send O masked)

(tile A O 2)
(tile B O 2)
(tile C O 2)
```

Il risultato restituito dalla Figura Scheme, con una piccola integrazione di colore, è rappresentato dalla figura 4.17.

Esercizio 25 *Scegli da* `http://members.cox.net/encyclpedia.html` *un insieme a tua scelta e riproducilo. Dopo aver esportato la figura ritoccala utilizzando The Gimp.*

Capitolo 5

EduKnoppix

Il software Dr. Geo non funziona in ambiente Windows e dunque per molti si pone il problema di come fare ad utilizzare il programma per le esercitazioni a casa o per qualsiasi altro uso. Come avrai notato, in allegato a questo libro c'è una copia del cdrom **EduKnoppix**. Eduknoppix è una **distribuzione di GNU/Linux**, contenuta in un solo cdrom, rivolta principalmente al mondo della Scuola.
Artefice principale e ideatore di eduKnoppix è il professor Maurizio Paolini dell'Università Cattolica di Brescia, il quale si avvale per la realizzazione della distribuzione dei contributi di diversi esponenti della comunità di sviluppo del software *libero*.
La distribuzione eduKnoppix è basata sulla distribuzione **Knoppix**, che l'ingegnere tedesco Klaus Knopper mantiene, la quale è stata a sua volta costruita a partire dalla distribuzione Debian (versioni testing/unstable).
Il funzionamento di eduKnoppix può avvenire su ogni personal computer con processore i86, indipendentemente dal sistema operativo che esso ospita, **effettuando un avvio da cdrom**. La procedura semplicissima che permette di utilizzare eduKnoppix è descritta nel paragrafo successivo.
Per sgombrare il campo dagli equivoci conviene subito precisare due fatti:

1. **eduKnoppix non si può utilizzare all'interno di un sistema operativo**: per citare un errore classico, **non** è possibile usare eduKnoppix entrando nel sistema operativo Windows e aprendo il cdrom.

2. l'utilizzo di eduKnoppix **non altera in alcun modo** il contenuto del disco rigido della macchina su cui esso viene lanciato: eduKnoppix infatti lavora utilizzando **esclusivamente** la memoria RAM del computer.

La filosofia alla base di eduKnoppix è di permettere, in particolare agli studenti, di poter familiarizzare con il sistema operativo GNU/Linux e con alcuni programmi utili per la didattica e di permettere loro di conseguire la Patente Europea del Computer (open-ECDL) su piattaforma *libera*.

5.1 Come ottenere eduKnoppix

Anche se ti è stata regalata una versione di eduKnoppix sappi che esistono vari modi per procurarsi il cdrom, tra cui:

- scaricarlo dal sito ufficiale `www.eduknoppix.org` o da uno dei suoi mirror;
- copiarlo da un amico;
- riceverlo in omaggio allegato alle riviste di software.

È importante ricordare che, trattandosi di *software libero*, tutti questi metodi sono legali.
Per chi dispone di una connessione Internet sufficientemente veloce, il metodo più semplice per ottenere eduKnoppix è scaricarlo dalla rete. Il file da scaricare è un file immagine, con nome ad esempio eduknoppix-1.9.6.iso, che ha una dimensione di circa 700 MB. Con una connessione a larga banda l'operazione richiede circa un paio di ore. Dopo aver salvato l'immagine sul computer è necessario trasferirla su un cdrom utilizzando un masterizzatore e un software per la masterizzazione.

5.2 Aiuto e Documentazione

Prima di passare alla descrizione dei diversi utilizzi di eduKnoppix, ricordiamo agli utilizzatori che:

1. **documentazione**: all'interno del cdrom è disponibile il manuale in linea *eduBook* e nel sito ufficiale del progetto eduKnoppix è a disposizione documentazione, sia didattica che tecnica, sull'utilizzo del cdrom;
2. **aiuto**: per problemi che non si riescono a risolvere è possibile chiedere aiuto nella mailing list degli utilizzatori di eduKnoppix o al tuo insegnante.

5.3 Configurazione di un computer per l'avvio di eduKnoppix

L'avvio di eduKnoppix si effettua semplicemente inserendo il cdrom nell'apposito cassetto, immediatamente dopo l'avvio del computer. Molto spesso i personal computer sono configurati in modo tale che la prima periferica ricercata all'avvio del sistema operativo sia il lettore di dischetti. Nel momento in cui non viene trovato alcun dischetto la seconda periferica ricercata è il lettore di cdrom. Se il lettore non contiene alcun cdrom allora il sistema

operativo viene lanciato a partire dal disco rigido. **Per configurazioni come questa l'avvio di EduKnoppix non richiede alcuna operazione accessoria!**
Se la configurazione di avvio del tuo computer non coincide con quella appena descritta, allora è indispensabile modificarla configurando il Setup (Bios). Per accedere al Bios è sufficiente premere un tasto, di solito F2 o CANC, subito dopo aver eseguito l'accensione del computer. Si deve avere l'accortezza di premere il tasto prima di udire il classico bip che si sente all'avvio del computer. A questo punto, comparirà una finestra (Bios Setup Utility) che occupa l'intero schermo con un menu. Qualsiasi modifica del Bios non verrà presa in considerazione se non quando uscirete da esso. Per uscire si hanno due scelte: "uscire senza salvare" oppure "uscire salvando" le modifiche.
Tra le indicazioni presenti sullo schermo dovrebbe comparire il menu Boot. Cliccando su questo menu si può modificare l'ordine in cui vengono lette le periferiche Boot Device al momento dell'avvio. Di solito si hanno diverse scelte: first boot device (prima periferica di avvio), second boot device (seconda perifierica di avvio), ecc. Noi consigliamo il seguente ordine:

- Prima: lettore di dischetti (removable devices)
- Seconda: lettore di CD (atapi cd drive)
- Terza:disco rigido (hard drive)

Realizzata la configurazione si esce, di solito premendo ESC. A quel punto il sistema vi chiederà se volete o meno salvare le modifiche e voi dovrete confermare. Vi è un modo diretto per effettuare la procedura di uscita premendo direttamente il tasto SAVE & EXIT. Appena usciti dal Bios udirete un bip e quindi il computer si riavvierà. Non ci sarà alcun bisogno di ripetere in futuro questa operazione e ogni volta che riavvierete il computer con il cd "live" all'interno del lettore di cdrom, esso verrà lanciato andando a lavorare utilizzando solo la memoria RAM. Chiaramente, se non inserite alcun cdrom il vostro sistema verrà avviato nella modalità abituale andando a leggere il contenuto del disco rigido.

5.4 Sistema Operativo e Interfaccia Grafica

Dopo essere riusciti ad avviare eduKnoppix, in pochi minuti ti troverai di fronte un sistema operativo completo di numerosi programmi.

5.4.1 GNU/Linux

Come ti avevamo anticipato eduKnoppix contiene un sistema operativo **GNU/Linux**, ossia un sistema ottenuto aggregando il kernel *Linux*, ideato da Linus Torvalds nel 1991, con i programmi elaborati a partire dal 1984

all'interno del *progetto GNU* di Richard Stallman. Il sistema operativo GNU/Linux è caratterizzato da un'architettura modulare ed essendo nato come clone del sistema operativo *Unix* risulta particolarmente adatto alla gestione di reti e di ambienti multiutente. Il fatto che GNU/Linux venga poi rilasciato con il codice sorgente aperto facilita enormemente il processo di correzione di errori (debugging) dei programmi in esso contenuti, con un conseguente aumento della sicurezza e dell'affidabilità del sistema nel suo complesso.

5.4.2 Interfaccia Grafica

La distribuzione eduKnoppix utilizza, come predefinita, un'interfaccia grafica a finestre di tipo **KDE** (K Desktop Environment) molto simile a quella utilizzata da altri sistemi proprietari come MS-Windows e Mac/OS. Per questo motivo, anche l'utilizzatore che proviene dall'uso di questi sistemi operativi, non dovrebbe incontrare difficoltà particolari nell'uso del sistema operativo attraverso questa interfaccia grafica.

5.5 Software di utilizzo comune

In eduKnoppix sono presenti numerosi programmi di uso comune. Ti ricordiamo, seguendo lo schema del menu, i seguenti che potrebbero risultare di tuo interesse:

- **Accessori**: il programma più importante di questa sezione è lo strumento di archiviazione **Ark**. Attraverso Ark è possibile comprimere e decomprimere file in diversi formati tra cui il `.zip` e il `.tgz`.

- **Impostazioni**: il programma più importante in questa sezione è il **Centro di Controllo** che permette di configurare i diversi componenti dell'interfaccia grafica KDE.

- **Knoppix**: in questa sezione si trovano diversi programmi per la configurazione del sistema. Di particolare interesse è la sezione Connessioni da cui è possibile configurare la connessione a Internet nelle situazioni più diverse (ADSL, Modem,...).

- **OpenOffice.org**: questa sezione contiene tutti i programmi della **suite office OpenOffice.org** che è nata come un clone di MS-Office.

- **Sistema**: in questa sezione sono disponibili diversi software utili per la gestione del sistema.

5.6 EduKnoppix e la Patente Europea

Una delle certificazioni informatiche maggiormente diffuse nel mondo della scuola è senza dubbio ECDL, acronimo di European Computer Driving License, forse più conosciuta con il nome di Patente Europea del Computer. Il conseguimento di ECDL avviene previo superamento di sette esami relativi ai seguenti moduli:

1. concetti di base
2. uso del computer e gestione dei file
3. editor di testo
4. foglio di calcolo
5. database
6. presentazioni
7. informazione e comunicazione

I contenuti e le competenze necessarie per poter superare i diversi esami sono spiegati dettagliatamente in un Sillabo disponibile al sito ufficiale della ECDL Foundation. I contenuti del Sillabo non fanno riferimento diretto ad una piattaforma informatica precisa ma nell'immaginario collettivo il riferimento rimane il sistema operativo Windows con la suite MS-Office.
L'associazione AICA, che da anni gestisce la rete di infrastrutture che provvedono la certificazione ECDL, offre, a partire dall'ottobre del 2003, la possibilità di conseguire questa certificazione ricorrendo **esclusivamente** a software *libero* in ambiente GNU/Linux. Questa certificazione prende il nome di **open-ECDL** e il titolo che viene rilasciato al termine del percorso è del tutto identico a quello che si ottiene su piattaforma MS-Windows. Gli abbinamenti tra moduli e software GNU/Linux sono i seguenti:

1. concetti di base
2. uso del computer e gestione dei file: ambiente KDE
3. editor di testo: OpenOffice.org Writer
4. foglio di calcolo: OpenOffice.org Calc
5. database: interfaccia OpenOffice.org verso un database
6. presentazioni: OpenOffice.org Impress
7. informazione e comunicazione: il navigatore e il client di posta Mozilla.

Tutti questi software sono contenuti in EduKnoppix. Maggiori dettagli sulle modalità del conseguimento della Patente Europea si possono ottenere al sito Aicanet.

Capitolo 6

Appendice A

In questa appendice sono riportati i **metodi di riferimento** utilizzabili per redigere uno script Guile per Dr. Geo.

`(getAbscissa punto)`
`punto`: punto su una curva
output: ascissa curvilinea del punto sulla curva. Il valore appartiene all'intervallo chiuso $[0, 1]$.

`(setAbscissa punto x)`
`punto`: punto su una curva
`x`: valore decimale nell'intervallo $[0, 1]$ che rappresenta l'ascissa del punto.

`(getCoordinates punto|vettore)`
`punto|vettore`: punto o vettore
output: lista contenente le coordinate del punto o del vettore

Esempio:

```
(define c (getCoordinates a1))
(define x (car c))
(define y (cadr c))
(+ (* x x) (* y y))
```

`(setCoordinates punto coordinate)`
`punto`: punto libero nel piano
`coordinate`: lista di numeri decimali

Esempio:

```
(define l (list 1.4 (random 5)))
(setCoordinate a1 l)
```

`(getSlope direzione)`
`direzione`: retta, semiretta, segmento o vettore.

output: pendenza relativa alla direzione

`(getNorm vettore)`
`vettore`: vettore
output: norma del vettore

`(getLength segmento)`
`segmento`: segmento
output: lunghezza del segmento

`(getAngle angolo)`
`angolo`: angolo geometrico
output: ampiezza in gradi dell'angolo

`(getCenter circonferenza|arco)`
`circonferenza|arco`: circonferenza o arco di circonferenza
output: lista contenente le coordinate del centro

Esempio:

```
(define c (getCenter a1))
(car c)
```

`(getRadius circonferenza|arco)`
`circonferenza|arco`: circonferenza o arco di circonferenza
output: raggio

`(getLength circonferenza|arco)`
`circonferenza|arco`: circonferenza o arco di circonferenza
output: lunghezza della circonferenza o dell'arco

`(getValue numero)`
`numero`: numero
output: valore del numero

`(setValue numero v)`
`numero`: numero
`v`: valore decimale

Esempio:

```
(define v (getValue a1))
(setValue a2 v)
```

`(move oggetto u)`
`oggetto`: oggetto geometrico
`u`: vettore

Esempio:

```
(define v (vector .1 0))
(move a1 v)
```

Capitolo 7

Appendice B

La definizione di oggetti in una Figura Scheme avviene attraverso dei prototipi. Tuttavia, prima di ogni definizione di un oggetto in una figura, questa deve essere stata precedentemente creata con il comando `new-figure`.

`(new-figure nomefigura)`
`nomefigura`: stringa di caratteri
output: la chiamata produce l'apertura di una figura vuota con nome `nomefigura`.

Punto

I prototipi relativi ai punti sono i seguenti:

`(Point nome free x y)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`x`: ascissa del punto
`y`: ordinata del punto
output: riferimento per un punto libero nel piano con coordinate iniziali `x` e `y`.
Esempio:`(define p1 (Point A free 1.2 3.2))`

`(Point nome on-curve linea x)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`linea`: riferimento ad una linea (retta, semiretta, segmento, ecc.)
`x`: ascissa curvilinea di un punto libero il cui valore appartiene all'intervallo $[0, 1]$
output: riferimento per un punto libero su una curva.
Esempio: `(Point M on-curve s1 0.5)`

`(Point nome middle-2pts p1 p2)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`p1`: riferimento ad un punto
`p2`: riferimento ad un punto
output: riferimento al punto medio tra due punti.

Esempio:

```
(lets Point "A" free 1 1)
(lets Point "B" free 4 4)
(Point "I" middle-2pts A B)
```

`(Point nome middle-segment s)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`s`: riferimento ad un segmento
output: riferimento al punto medio di un segmento
Esempio: `(Point L middle-segment s)`

`(Point nome intersection l1 l2)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`l1`: riferimento ad una linea
`l2`: riferimento ad una linea
output: riferimento al punto di intersezione di due linee
Esempio: `(Point I intersection line segment)`

`(Point nome intersection2 l1 l2)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`l1`: riferimento ad una linea
`l2`: riferimento ad una curva
output: riferimento al secondo punto d'intersezione di due curve quando una delle due è un arco o una circonferenza
Esempio: `(Point I intersection2 line circle)`

Retta

`(Line nome 2points p1 p2)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`p1`: riferimento ad un punto
`p2`: riferimento ad un punto
output: riferimento ad una retta passante per due punti

Esempio:

```
(lets Point "A" free 0 0)
(lets Point "M" free 1 2)
(Line "" 2points A M)
```

`(Line nome parallel p d)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`p`: riferimento ad un punto
`d`: riferimento ad una direzione (retta, segmento, vettore ...)

output: riferimento ad una retta parallela alla direzione `d` e passante per `p`.

Esempio:

```
(lets Point "A" free 1 5)
(lets Line "d1" parallel A d)
```

`(Line nome orthogonal p d)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`p`: riferimento ad un punto
`d`: riferimento ad una direzione (retta, segmento, vettore ...)
output: riferimento ad una retta perpendicolare alla direzione `d` e passante per `p`.

Esempio:

```
(lets Point "A" free 1 5)
(lets Line "d1" orthogonal A d)
```

Semiretta

`(Ray nome 2points o p)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`o`: riferimento ad un punto origine della semiretta
`p`: riferimento ad un punto appartenente alla semiretta
output: riferimento ad una semiretta definita dall'origine e un punto.

Esempio:

```
(lets Point "A" free 1 5)
(lets Point "O" free 0 0)
(lets Ray "dd1" 2points A 0)
```

Segmento

`(Segment nome extremities p1 p2)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`p1`: riferimento ad un punto estremo del segmento
`p2`: riferimento ad un punto estremo del segmento
output: riferimento ad un segmento definito dai suoi estremi.

Esempio:

```
(lets Point "A" free 1 5)
(lets Point "B" free 10 4)
(lets Segment "" extremities A B)
```

Circonferenza

`(Circle nome 2points c p)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`c`: riferimento ad un punto: centro della circonferenza
`p`: riferimento ad un punto appartenente alla circonferenza
output: riferimento ad una circonferenza definita da centro e punto ad essa appartenente.

Esempio:

```
(lets Point "A" free 1 5)
(lets Point "B" free 10 4)
(lets Circle "C1" 2points A B)
```

`(Circle nome center-radius c r)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`c`: riferimento ad un punto: centro della circonferenza
`r`: riferimento ad un valore numerico: raggio della circonferenza
output: riferimento ad una circonferenza definita da centro e raggio.

Esempio:

```
(lets Point "A" free 1 5)
(lets Numeric "r" free 10 4)
(lets Circle "C1" center-radius A r)
```

`(Circle nome center-segment c s)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`c`: riferimento ad un punto: centro della circonferenza
`s`: riferimento ad un segmento la cui lunghezza definisce il raggio della circonferenza
output: riferimento ad una circonferenza definita da centro e raggio.

Esempio:

```
(lets Point "A" free 1 5)
(lets Point "B" free 10 4)
(lets Segment "s" extremities A B)
(lets Circle "C1" center-segment A s)
```

Arco

`(Arc nome 3points p1 p2 p3)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`p1`: riferimento ad un punto primo estremo del'arco
`p2`: riferimento ad un punto centro dell'arco

`p3`: riferimento ad un punto secondo estremo dell'arco
output: riferimento ad un arco di circonferenza definito dai suoi estremi e da un punto.

Esempio:

```
(lets Point "A" free 1 5)
(lets Point "B" free 0 5)
(lets Point "C" free -1 -2)
(lets Arc "arc" 3points A B C)
```

Poligono

`(Polygon nome npoints args)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`args`: una lista di riferimenti a punti che rappresentano i vertici del poligono
output: riferimento ad un poligono definito dai suoi vertici.

Esempio:

```
(lets Point "A" free 1 1)
(lets Point "B" free 1 5)
(lets Point "C" free 5 1)
(lets Point "D" free 5 5)
(lets Polygon "quad" npoints A B C D)
```

Trasformazioni geometriche

I prototipi relativi alle trasformazioni geometriche permettono di eseguire trasformazioni geometriche di diversi oggetti. Essi si applicano in riferimento agli oggetti punto, segmento, retta, semiretta, vettore, circonferenza, arco e poligono.
`(TipoOggetto nome rotation oggetto centro angolo)`
`TipoOggetto`: Point, Segment, Line, Ray, Vector, Circle, Arc, Polygon
`nome`: stringa di caratteri che designa il nome dell'oggetto
`oggetto`: riferimento all'oggetto da trasformare
`centro`: riferimento ad un punto: il centro di rotazione
`angolo`: riferimento ad un valore: l'angolo di rotazione
output: riferimento all'oggetto trasformato.
Esempio: `(lets Point I1 rotation I C a)`

`(TipoOggetto nome scale oggetto centro k)`
`TipoOggetto`: Point, Segment, Line, Ray, Vector, Circle, Arc, Polygon
`nome`: stringa di caratteri che designa il nome dell'oggetto
`oggetto`: riferimento all'oggetto da trasformare
`centro`: riferimento ad un punto: il centro di omotetia

`k`: riferimento ad un valore: rapporto di omotetia
output: riferimento all'oggetto trasformato.
Esempio: `(lets Polygon P1 scale P C k1)`

`(TipoOggetto nome symmetry oggetto centro)`
`TipoOggetto`: Point, Segment, Line, Ray, Vector, Circle, Arc, Polygon
`nome`: stringa di caratteri che designa il nome dell'oggetto
`oggetto`: riferimento all'oggetto da trasformare
`centro`: riferimento ad un punto: il centro di simmetria
output: riferimento all'oggetto trasformato.
Esempio: `(lets Segment S1 symmetry S C)`

`(TipoOggetto nome reflexion oggetto asse)`
`TipoOggetto`: Point, Segment, Line, Ray, Vector, Circle, Arc, Polygon
`nome`: stringa di caratteri che designa il nome dell'oggetto
`oggetto`: riferimento all'oggetto da trasformare
`asse`: riferimento ad una retta: l'asse di riflessione
output: riferimento all'oggetto trasformato.
Esempio: `(lets Polygon P1 reflexion P d1)`

`(TipoOggetto nome translation oggetto vettore)`
`TipoOggetto`: Point, Segment, Line, Ray, Vector, Circle, Arc, Polygon
`nome`: stringa di caratteri che designa il nome dell'oggetto
`oggetto`: riferimento all'oggetto da trasformare
`vettore`: riferimento ad un vettore: il vettore di traslazione
output: riferimento all'oggetto trasformato.
Esempio: `(lets Circle C1 translation C v)`

Luogo geometrico

`(Locus nome 2points m c)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`m`: riferimento ad un punto mobile su una curva
`c`: riferimento ad un punto fisso dipendente dal punto `m`
output: riferimento ad un luogo.
Esempio: `(Locus luogo 2points M I)`

Vettore

`(Vector nome 2points o e)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`o`: riferimento ad un punto: origine del vettore
`e`: riferimento ad un punto: estremo del vettore
output: riferimento ad un vettore.

Esempio:

```
(lets Point "B" free 0 5)
(lets Point "C" free -1 -2)
(Vector "" 2points C B)
```

Numeri

`(Numeric nome free x y v)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`x`,`y`: le coordinate che determinano la posizione del numero
`v`: il valore del numero
output: riferimento a un numero.
Esempio: `(lets Numeric pi free 5 5 3.5)`

`(Numeric nome segment-length x y s)` `nome`: stringa di caratteri che designa il nome dell'oggetto
`x`,`y`: le coordinate che determinano la posizione del numero
`s`: riferimento ad un numero: la lunghezza di un segmento
output: riferimento alla lunghezza di un segmento.
Esempio: `(lets Numeric l segment-length 5 5 s)`

`(Numeric nome vector-norm x y v)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`x`,`y`: le coordinate che determinano la posizione del numero
`v`: riferimento ad un vettore
output: riferimento alla norma di un vettore.
Esempio: `(lets Numeric l vector-norm 5 5 v)`

`(Nombre nome point-circle x y p c)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`x`,`y`: le coordinate che determinano la posizione del numero
`p`: riferimento ad un punto
`c`: riferimento ad una circonferenza
output: riferimento al valore che esprime la distanza tra punto e circonferenza.
Esempio: `(lets Numeric l point-circle 5 5 P c)`

`(Numeric nome point-line x y p d)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`x`,`y`: le coordinate che determinano la posizione del numero
`p`: riferimento ad un punto
`d`: riferimento ad una retta
output: riferimento al valore che esprime la distanza tra punto e retta.
Esempio: `(lets Numeric d point-line 5 5 M d1)`

`(Numeric nome point-point x y p1 p2)`
`nome`: stringa di caratteri che designa il nome dell'oggetto

`x`,`y`: le coordinate che determinano la posizione del numero
`p1`: riferimento ad un punto
`p2`: riferimento ad un punto
output: riferimento al valore che esprime la distanza tra i due punti.
Esempio: `(lets Numeric d point-point 5 5 A B)`

`(Numeric nome circle-length x y c)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`x`,`y`: le coordinate che determinano la posizione del numero
`c`: riferimento ad una circonferenza
output: riferimento alla lunghezza della circonferenza.
Esempio: `(lets Numeric p circle-length 5 5 circ)`

Angolo

`(Angle nome geometric A B C)`
`nome`: stringa di caratteri che designa il nome dell'oggetto
`A`: riferimento a un punto
`B`: riferimento a un punto: vertice dell'angolo
`C`: riferimento a un punto
output: riferimento ad un angolo geometrico.
Esempio: `(lets Angle a geometric A B C)`

Attributi

Per modificare gli attributi di un oggetto creato in precedenza si utilizza un sistema di messaggi che vengono inviati direttamente al prototipo che rappresenta l'oggetto in questione. Le modifiche agli attributi si eseguono quindi a posteriori della costruzione dei diversi oggetti.

```
(send oggetto color valore)
```

`oggetto`: riferimento al simbolo di un oggetto
`valore`: un colore tra i seguenti: black, dark-grey, grey, white, dark-green, green, dark-blue, bleu, red, yellow
Esempio:

```
(lets Point "A" free 1 2)
(send A color green)
```

`(send linea thickness valore)`
`linea`: riferimento ad una linea (retta, semiretta, circonferenza, luogo, ecc.)
`valore`: lo spessore i cui valori possibili sono: dashed, normal, large
Esempio:

```
(lets Point "A" free 1 2)
(lets Point "O" free 0 0)
```

```
(lets Line "d" 2points A B)
(send d thickness dashed)
```

```
(send punto size valore)
```
punto: riferimento ad un punto
`valore`: la dimensione di un punto i cui valori possibili sono: small, normal, large
Esempio:

```
(lets Point "A" free 1 2)
(send A size small)
```

```
(send punto shape valore)
```
`punto`: riferimento ad un punto
`valore`: la forma di un punto i cui valori possibibili sono: round, cross, round-empty, rec-empty
Esempio:

```
(lets Point "A" free 1 2)
(send A shape rond)
```

```
(send oggetto masked)
```
`oggetto`: riferimento ad un oggetto nascosto
Esempio:

```
(lets Point "A" free 1 2)
(send A masked)
```

Ringraziamenti

Ringrazio Lucia Gecchelin per aver letto e migliorato il manoscritto.

Capitolo 8

GNU Free Documentation License

GNU Free Documentation License
Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free

program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that

the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty

Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent

copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections

and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list

of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for

verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements",

"Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation;

```
    with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
    A copy of the license is included in the section entitled "GNU
    Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts,
replace the "with...Texts." line with this:

    with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other
combination of the three, merge those two alternatives to suit the
situation.

If your document contains nontrivial examples of program code, we
recommend releasing these examples in parallel under your choice of
free software license, such as the GNU General Public License,
to permit their use in free software.
```

Bibliografia

[1] C.B. Boyer, *Storia della Matematica*, Mondadori, Milano, 1980.

[2] A. Centomo & H. Fernandes, *Manuale di Dr. Geo*, Ofset, Mont de Marsan, 2004.

[3] A. Centomo, Figure Scheme per Dr. Geo, Lettera Matematica Pristem, n. 55, 2005.

[4] A. Centomo, Dr. Geo e la Geometria tolemaica, Lettera Matematica Pristem, n. 49, 2003.

[5] M. Dedò, *Trasformazioni Geometriche*, Zanichelli Decibel, Padova, 1996.

[6] G. A. Edgar, *Measure, Topology and Fractal Geometry*, Springer-Verlag, New York, 1992.

[7] M. Fellisen R. B. Findler M. Flatt S. Krishnamurti, *How to design Programming*, MIT Press, Cambridge, 2003.

[8] B. Grünbaum G.C. Shephard, *Tilings and Patterns*, W. H. Freeman and Company, New York, 1987.

[9] M. Livio, *La sezione aurea*, Rizzoli, Milano, 2003.

[10] B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, San Francisco, 1982.

[11] Platone, *Tutti gli scritti*, G. Reale (a cura di), Editore Rusconi, Milano, 1992.

[12] P. Prusinkiewicz A. Lindermayer, *The algorithmic beauty of plants*, Springer-Verlag, New York, 1990.

[13] M. Senechal, *Quasicrystals and Geometry*, Cambridge University Press, Cambridge, 1996.

[14] G.J. Toomer, *Ptolemy's Almagest*, Gerald Duckworth & Co.Ltd., London, 1984.

[15] P. Zellini, *Gnomon*, Adelphi, Milano, 1999.

Indice

Licenza