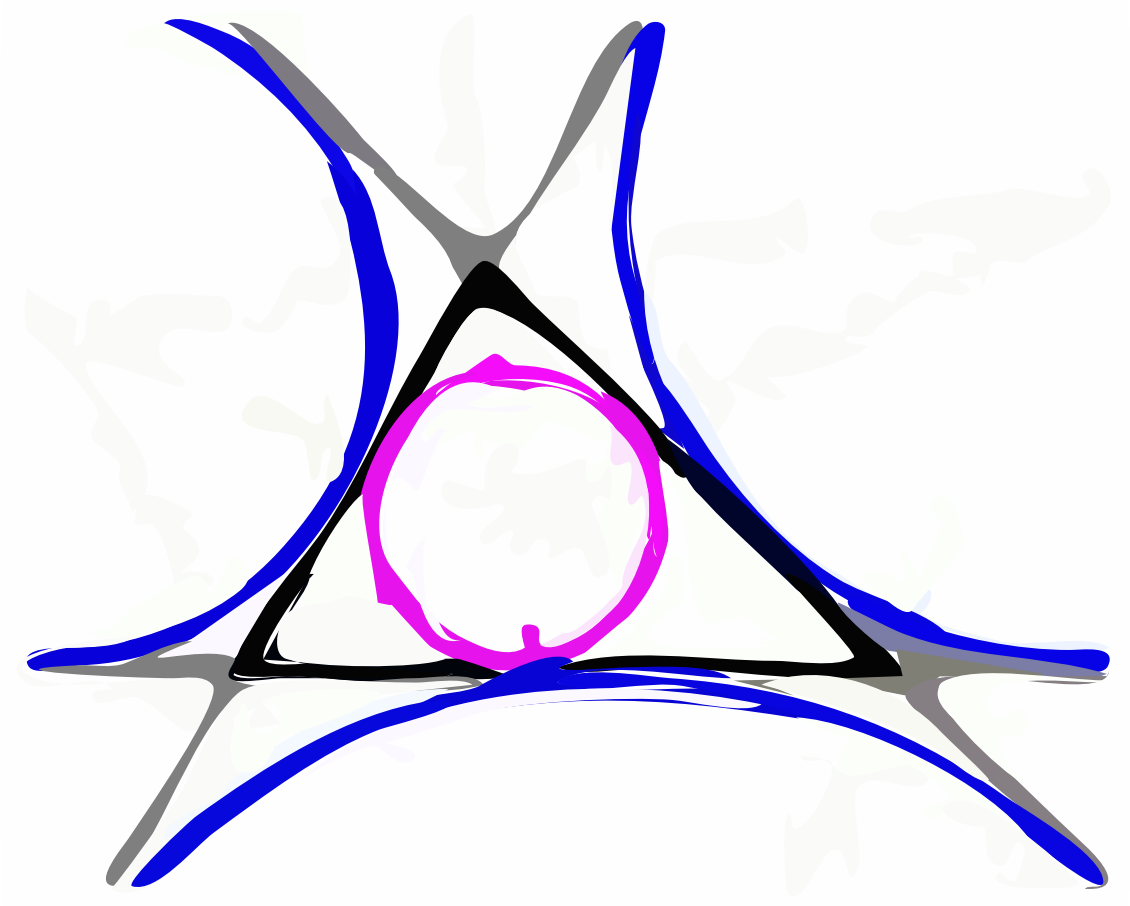
Geometria euclidea con Eukleides

Articolo estratto dall'opera «Informatica per sopravvivere»

Massimo Piai ⟨*pxam67 (at) virgilio·it*⟩

2006.02.18



Massimo Piai è un matematico appassionato di informatica, che ha trovato nel software libero e nella libertà delle informazioni l'unica possibilità di sviluppare tale passione. I suoi campi di interesse attuali sono la matematica e le scienze, la diffusione della Cultura Informatica, la didattica e la pedagogia.

Il presente lavoro è stato realizzato utilizzando Alml, il sistema di composizione SGML realizzato da Daniele Giacomini per la gestione dei suoi *Appunti di informatica libera*.

Indice generale

Introduzione

Eukleides[1] è un mini-linguaggio grafico per la geometria euclidea. Associati a Eukleides esistono due programmi principali:

- '**`eukleides`**', un compilatore che permette il disegno di figure geometriche all'interno di documenti TeX oppure LaTeX, utile anche per convertire tali figure in formato EPS o in altri formati grafici vettoriali;
- '**`xeukleides`**', un programma frontale per X che consente la creazione di figure geometriche interattive, utile anche per la modifica e la messa a punto del codice sorgente Eukleides.

Eukleides è stato progettato in modo da essere simile al linguaggio naturale della geometria euclidea elementare. Molto spesso è possibile evitare completamente l'utilizzo delle coordinate cartesiane. Nelle sezioni 4 e 5 vengono mostrati molti esempi che illustrano questa e altre caratteristiche del linguaggio.

Il linguaggio Eukleides e i programmi '**`eukleides`**' e '**`xeukleides`**' sono opera di Christian Obrecht, ⟨*obrecht (ad) altern·org*⟩.

Al momento della presente stesura la versione di riferimento dei programmi è la 0.9.2; Christian Obrecht sta lavorando a una ristrutturazione piuttosto completa del software, che includerà molte interessanti novità, fra cui la localizzazione del linguaggio utilizzato per le parole chiave.

---

[1] **Eukleides** GNU GPL

Guida elementare

In questa sezione vengono illustate alcune delle possibilità del programma '**eukleides**', degli script '**euk2eps**' e '**euk2edit**' e del programma '**xeukleides**'.

1.1 Disegno di un triangolo

Il linguaggio Eukleides è stato progettato per essere simile al linguaggio tradizionale della geometria piana euclidea. Ad esempio, per disegnare un triangolo è sufficiente preparare un file contenente il listato 1.2: se il file si chiama '`triangle.euk`', ecco come si potrebbe procedere:

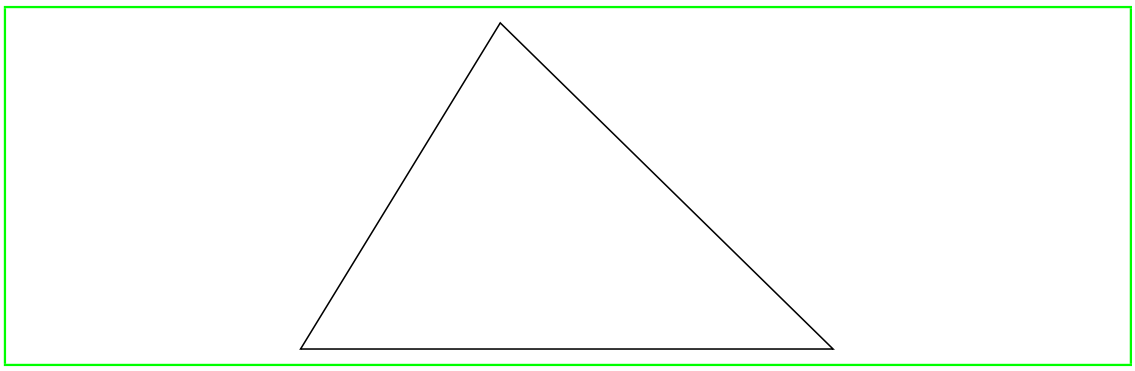
```
$ eukleides triangle.euk [ Invio ]

  % Generated by eukleides 0.9.2
  \psset{linecolor=black, linewidth=.5pt, arrowsize=2pt 4}
  \psset{unit=1.0000cm}
  \pspicture*(-2.0000,-2.0000)(8.0000,6.0000)
  \pspolygon(0.0000,0.0000)(6.0000,0.0000)(2.2500,3.6742)
  \endpspicture


$
```

L'output così generato può essere copiato così com'è in un sorgente TeX oppure LaTeX, ricordandosi però di includere anche il pacchetto '**pstricks**'. Il documento generato conterrà la figura 1.3.

Listato 1.2. Eukleides: esempio elementare.

```
A B C triangle
draw(A, B, C)
```

Figura 1.3.



Per ottenere più rapidamente una figura da visualizzare si può ricorrere allo script '**euk2eps**' il quale genera un file in formato EPS che a sua volta è visualizzabile tramite (ad esempio) GV. Esiste inoltre lo script '**euk2edit**' il quale, appoggiandosi a Pstoedit, consente la conversione in vari formati grafici vettoriali come ad esempio quello utilizzato dal programma XFig; per esempio, per poter successivamente modificare con XFig l'immagine è sufficiente procedere come segue:

```
$ euk2edit triangle.euk fig [ Invio ]
```

```
This is eukleides version 0.9.2
Copyright (c) Christian Obrecht 2000-2002
==> Warning: BoundingBox not found!
pstoedit: version 3.33 / DLL interface 108 (build Jul 29 2004 - release build) :
 Copyright (C) 1993 - 2003 Wolfgang Glunz
Warning: some types of raster images in the input file cannot be converted if th
e output is sent to standard output
Warning: Level 2 version of image and imagemask not supported for this backend (
due to lack of support for FILE files)
Interpreter finished. Return status 0
```

$ **cat triangle.fig** [*Invio*]

```
#FIG 3.2
Portrait
Flush left
Inches
Letter
100.00
Single
0
1200 2
# polyline
2 1 0 1 0 0 999 0 -1 4.0 0 0 0 0 0 5
        2675 3166 2143 4033 4978 4033 3206 2298 2675 3166
```

$

Spesso risulta più comodo avere un singolo file che contenga sia testo che figure; in tal caso è possibile utilizzare l'opzione '**-f**' e i commenti speciali '**%--eukleides**' e '**%--end**' nel sorgente TeX oppure LaTeX. Ad esempio:

$ **cat triangle.etex** [*Invio*]

```
\input pstricks
This is a scalene triangle:\par
%--eukleides
A B C triangle
draw(A, B, C)
%--end
\bye
```

$ **eukleides -f triangle.etex > triangle.tex** [*Invio*]

$ **cat triangle.tex** [*Invio*]

```
\input pstricks
This is a scalene triangle:\par
% Generated by eukleides 0.9.2
\psset{linecolor=black, linewidth=.5pt, arrowsize=2pt 4}
\psset{unit=1.0000cm}
```

```
\pspicture*(-2.0000,-2.0000)(8.0000,6.0000)
\pspolygon(0.0000,0.0000)(6.0000,0.0000)(2.2500,3.6742)
\endpspicture
% End of figure
\bye
```

```
$
```

Dal file ‘`triangle.tex`’ è quindi possibile ottenere un file in formato DVI e quindi uno in formato PostScript:

```
$ tex triangle.tex [ Invio ]
```

```
This is TeX, Version 3.14159 (Web2C 7.4.5)
(./triangle.tex (/usr/share/texmf/tex/generic/pstricks/pstricks.tex
‘PSTricks’ v97 patch 14  <1999/12/23> (tvz)
(/usr/share/texmf/tex/generic/pstricks/pstricks.con)) [1] )
Output written on triangle.dvi (1 page, 740 bytes).
Transcript written on triangle.log.
```

```
$ dvips -o triangle.ps triangle [ Invio ]
```

```
This is dvips(k) 5.92b Copyright 2002 Radical Eye Software (www.radicaleye.com)
’ TeX output 2006.02.05:2049’ -> triangle.ps
<texc.pro><pstricks.pro><pst-dots.pro><f7b6d320.enc><texps.pro><special.pro>
. <cmr10.pfb>[1]
```

```
$
```

1.2 Ulteriori informazioni sui triangoli. Programma frontale interattivo

Il programma per X ‘**`xeukleides`**’ è un programma frontale interattivo per il linguaggio Eukleides. Ha due modalità di funzionamento:

1. modalità di ***modifica***,
2. modalità di ***presentazione***.

Volendo avviare ‘**`xeukleides`**’ e subito modificare il file di cui alla sezione 1.1, si digiti:

```
$ xeukleides triangle.euk [ Invio ]
```

Volendo invece avviare il programma in modalità di presentazione, è sufficiente aggiungere l’opzione ‘**`-V`**’ alla riga di comando. Per alternare fra le due modalità si utilizi il tasto [*Esc*] (figura 1.10). Siccome la modalità di modifica è realizzata grazie alle funzionalità per la modifica del testo offerte dalle librerie GTK+, ‘**`xeukleides`**’ possiede tutte le caratteristiche normalmente offerte da un semplice *editor* di testo; la tabella 1.11 elenca alcune utili scorciatoie.

Figura 1.10. '**xeukleides**': le due modalità di funzionamento (modifica e presentazione).

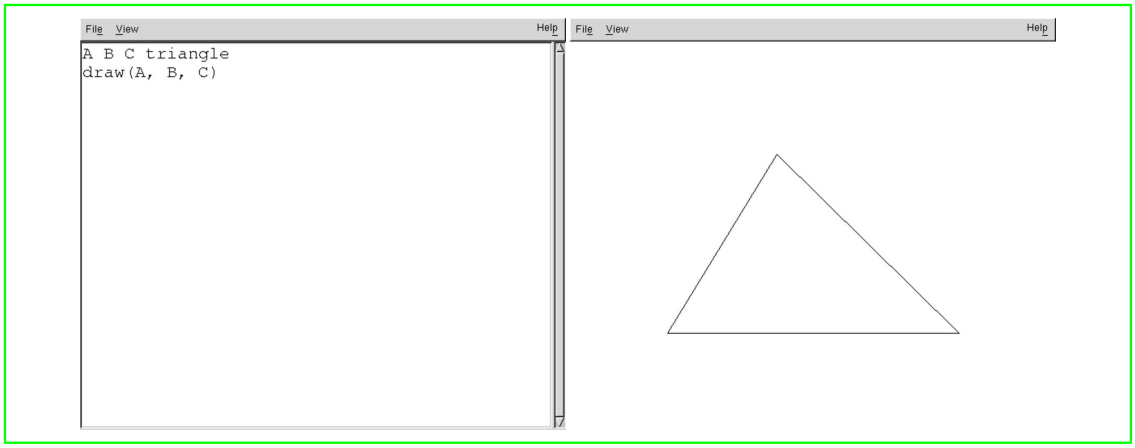
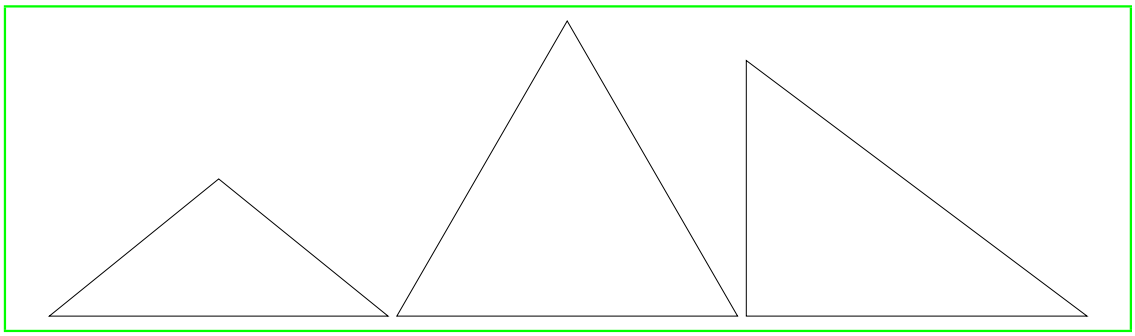


Tabella 1.11. '**xeukleides**': comandi utilizzabili nella modalità di modifica.

Comando	Descrizione
[*Alt F*]	Muove il cursore in avanti di una parola
[*Alt B*]	Muove il cursore indietro di una parola
[*Alt D*]	Cancella il testo fino alla fine della parola corrente
[*Ctrl W*]	Cancella il testo fino all'inizio della parola corrente
[*Ctrl U*]	Cancella il contenuto della riga corrente
[*Ctrl K*]	Cancella il testo fino alla fine della riga corrente
[*Ctrl X*]	Taglia il testo selezionato
[*Ctrl C*]	Copia il testo selezionato
[*Ctrl V*]	Incolla il testo selezionato

Il triangolo definito dal comando '**A B C triangle**' è un ***triangolo scaleno ottimale*** (ossia: un triangolo acutangolo la cui forma si discosta il più possibile sia da quella di un triangolo isoscele che da quella di un triangolo rettangolo). È ovviamentre possibile disegnare altri tipi di triangolo: il linguaggio Eukleides permette di definire i triangoli in molti modi; per esempio, la figura 1.12 illustra ciò che si ottiene sostituendo il comando '**triangle**' con il comando '**isosceles**' oppure '**equilateral**' oppure '**right**'.

Figura 1.12. Eukleides: triangolo iscoscele, triangolo equilatero, triangolo rettangolo.

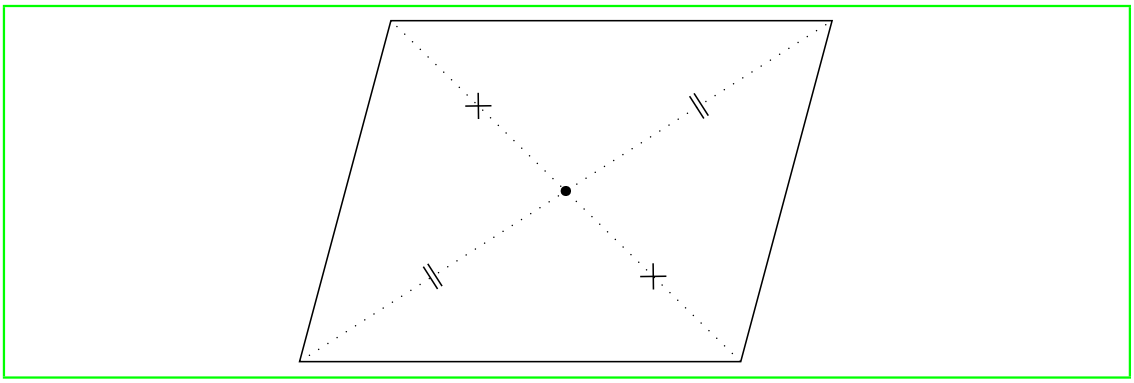


I suddetti comandi possono eventualmente accettare dei parametri opzionali, ad esempio '**A B C triangle(6, 5, 4)**' definisce un trinagolo ABC tale che i lati AB, BC e AC misurino rispettivamente 6 cm, 5 cm e 4 cm. Altro esempio: '**A B C isosceles(6, 50:)**' definisce un triangolo isoscele ABC tale che la base AB misuri 6 cm e gli angoli alla base misurino entrambi 50°.

1.3 Una proprietà dei parallelogrammi

In questa sezione si mostra come generare la figura 1.13.

Figura 1.13. Eukleides: le diagonali di un parallelogrammo si bisecano vicendevolmente.



Per prima cosa è necessario definire il parallelogrammo e il suo centro:

```
1  A B C D parallelogram
2  O = barycenter(A, B, C, D)
```

poi si possono disegnare entrambi:

```
3  draw(A, B, C, D) ; draw(O)
```

poi si disegnano le diagonali:

```
4  draw(segment(A, C), dotted)
5  draw(segment(B, D), dotted)
```

e infine si marcano le semidiagonali con dei doppi tratti:

```
6  mark(segment(O, A), double)
7  mark(segment(O, C), double)
```

e delle croci:

```
8  mark(segment(O, B), cross)
9  mark(segment(O, D), cross)
```

1.4 Una proprietà dei triangoli. Geometria interattiva

In questa sezione verrà introdotto un esempio di figura interattiva, tramite la quale si «dimostrerà» che un triangolo inscritto in una semicirconferenza è necessariamente rettangolo. Nella figura sarà possibile spostare il vertice che giace sulla semicirconferenza utilizzando i tasti [*freccia destra*] e [*freccia sinistra*].

Per prima cosa si definiscono i punti A e B e la circonferenza C di diametro AB, con i seguenti comandi:

```
1  A = point(0, 0) ; B = point(6, 0)
2  C = circle(A, B)
```

I punti A e B sono definiti tramite le loro coordinate cartesiane.[1] Successivamente si definisce una ***variabile interattiva***, ***t***:

```
3  t interactive(60, -2, 0, 180, "A", right)
```

Ciò significa che il valore iniziale di ***t*** è 60, il valore minimo è zero e il valore massimo è 180. In modalità di presentazione, ogniqualvolta si preme uno dei tasti [*freccia destra*] o [*freccia sinstra*] il valore `-2` viene addizionato o sottratto (rispettivamente) alla variabile. Indicando '**`up`**' al posto di '**`right`**' si associa la variabile ai tasti [*freccia su*] e [*freccia giù*]. Infine, '**`"A"`**' indica lo ***stato*** interno corrispondente alla variabile.[2]

Successivamente si definisce un punto M sulla circonferenza C:

```
4  M = point(C, t:)
```

Il secondo parametro è seguito dal simbolo di due punti ('**`:`**') per indicare che si tratta di un ***parametro angolare***: la variabile ***t*** corrisponde all'***argomento***[3] (seconda coordinata polare) del punto M rispetto al polo posto nel centro di C.

Dopodiché si disegna la semicirconferenza superiore di C e il triangolo ABM inscritto:

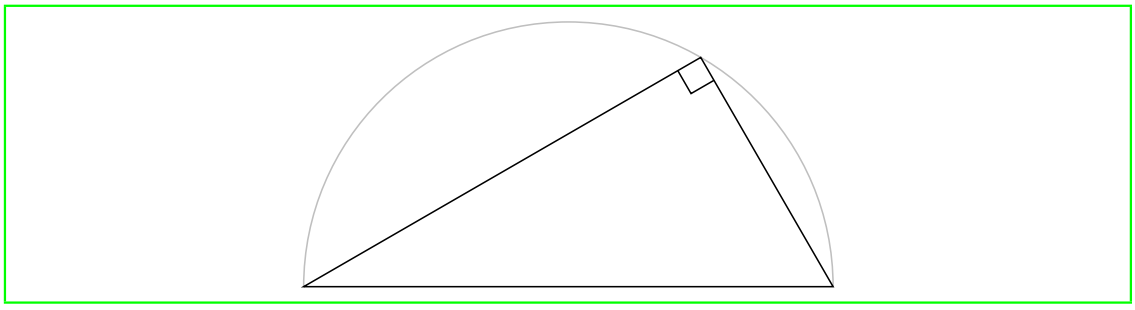
```
5  color(lightgray)
6  draw(C, 0:, 180:)
7  color(black)
8  draw(A, B, M)
```

Infine si marca l'angolo con vertice in M come retto:

```
9  mark(A, M, B, right)
```

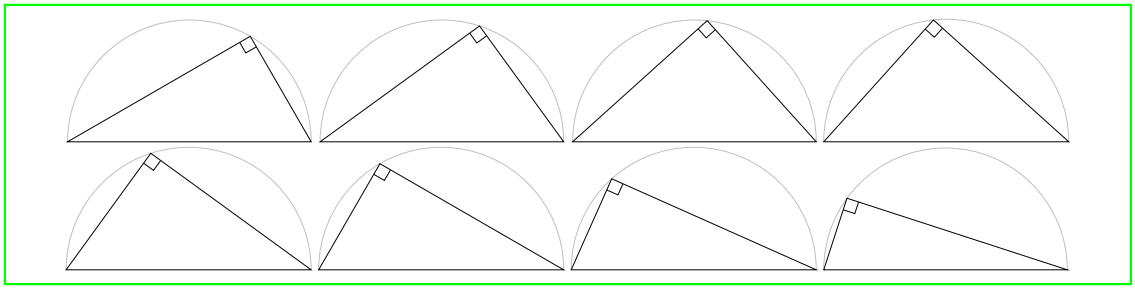
La figura 1.24 illustra il risultato finale.

Figura 1.24. Eukleides: un triangolo inscritto in una semicirconferenza è necessariamente un triangolo rettangolo.



È ora possibile interagire con la figura, grazie al programma '**`xeukleides`**', utilizzando i tasti freccia, e constatare infine graficamente la validità della congettura geometrica enunciata all'inizio (figura 1.25).

Figura 1.25. Eukleides: figure generate interattivamente tramite '**xeukleides**'.



Se si preme il tasto [*F1*] in modalità di presentazione si «cattura» il valore modificato della variabile interattiva: esso costituirà il nuovo valore iniziale della variabile, e il programma passa automaticamente alla modalità di modifica.

Il codice scritto per '**xeukleides**' può essere utilizzato come sorgente per '**eukleides**': le eventuali variabili interattive assumeranno staticamente il loro valore iniziale.

---

[1] Salvo quando diversamente specificato, la figura viene disegnata all'interno di un riquadro il cui vertice inferiore sinistro ha coordinate (-2;-2) e quello superiore destro coordinate (8;6).

[2] Quando si entra in modalità di presentazione il programma è nello stato "A"; essendoci 26 lettere da "A" a "Z" si possono definire 52 variabili interattive per la medesima figura (ogni stato può essere associato a due variabili, una per ciascuna coppia di tasti freccia); per cambiare stato basta premere il corrispondente tasto alfabetico.

[3] in gradi

Guida di riferimento

In questa sezione si descrive sistematicamente il linguaggio Eukleides.

2.1 Sintassi

Un ***file sorgente*** Eukleides può contenere:

- ***commenti***, ossia tutti i caratteri dal simbolo '`%`' alla fine della riga;
- ***assegnamenti semplici***, ossia un nome di variabile seguito da un simbolo '`=`' e da un'espressione;
- ***assegnamenti multipli***, ossia una sequenza di nomi di variabile, separata da spazi, seguita (senza '`=`') da un comando di assegnamento multiplo (assegnamento di un triangolo, di un poligono, di intersezione, ecc.);
- ***assegnamenti interattivi***, ossia un nome di variabile seguito dalla parola chiave '`interactive`' assieme ad alcuni parametri;
- ***comandi grafici***, ossia comandi per l'impostazione dei parametri, comandi di disegno oppure comandi di tracciamento.

I nomi di variabile sono alfanumerici e sensibili alla differenza tra le lettere maiuscole e minuscole; il primo carattere **deve** essere una lettera. Le variabili possono contenere vari tipi di oggetto: numeri, vettori, rette, segmenti, circonferenze, sezioni coniche. Una singola riga del sorgente può contenere diversi comandi o assegnamenti (separati da '`;`').

In questa sezione varranno alcune notazioni convenzionali nella descrizione dei parametri dei comandi:

- x, y, z: numeri;
- a, b: parametri angolari;
- A, B, C, D, E, F, G: punti;
- u, v: vettori;
- l, l': rette;
- s: un segmento;
- c, c': circonferenze;
- sc: una sezione conica;
- str: una stringa;
- f, f': indicatori.

Un ***parametro angolare*** è un'espressione numerica seguita dal simbolo '`:`' (per indicare ***gradi sessagesimali***) oppure dal simbolo '`<`' (per indicare ***gradi radianti***). Un stringa è una singola riga di testo fra doppi apici.[1]

I parametri opzionali verranno indicati fra parentesi quadre.

È inoltre opportuno fare alcune precisazioni di carattere terminologico.
Nel seguito si utilizzerà la parola «ascissa» in diversi contesti: il significato probabilmente più comune è quello di «prima coordinata in un piano cartesiano», ma il significato più generale è quello di «numero che indica (univocamente a meno di casi singolari) la posizione di un punto su una curva»; in quest'ultima definizione rientra in realtà anche il primo caso (se si identifica l'ascissa del punto con l'ascissa della sua proiezione sull'asse delle ascisse, e si considera che sull'asse delle ascisse viene fissata per l'appunto convenzionalemnte un'ascissa), nonché il caso della rappresentazione parametrica di una curva piana.
Un'altra osservazione è opportuna riguardo al termine «argomento»: in un lavoro di carattere informatico si tenderebbe ad attribuire ad esso il significato di «parametro attuale passato a una procedura o funzione», e talvolta anche nella presente sezione varrà tale significato; però, parlando di geometria, esiste un'altra possibilità, ossia quella di «seconda coordinata in un piano su cui sia stato fissato un sistema di riferimento polare».
Si ritiene che il contesto dovrebbe chiarire implicitamente il significato adottato di volta in volta; in caso contrario si specificherà esplicitamente.

2.2 Valori numerici

Sono disponibili le usuali operazioni matematiche, mediante i simboli: '**(**', '**)**', '**+**', '**-**', '*****', '**/**', '**^**' (elevamento a potenza).

Le funzioni numeriche disponibili sono: '**abs()**', '**sqrt()**', '**exp()**', '**ln()**', '**sin()**', '**cos()**', '**tan()**', '**asin()**', '**acos()**', '**atan()**', '**deg()**' (per convertire da radianti a gradi...), '**rad()**' (... e viceversa).

È disponibile la costante '**pi**'.

La tabella 2.1 riepiloga le ulteriori funzioni che restituiscono valori numerici.

Tabella 2.1. Eukleides: funzioni che restituiscono valori numerici.

Funzione	Note esplicative
`abscissa(`***A***`)` oppure `abscissa(`***u***`)`	Ascissa (prima coordinata cartesiana) del punto o dell'estremo del vettore.
`ordinate(`***A***`)` oppure `ordinate(`***u***`)`	Ordinata (seconda coordinata cartesiana) del punto o dell'estremo del vettore.
`distance(`***A***`, `***B***`)` oppure `distance(`***A***`, `***l***`)`	Distanza fra due punti, oppure fra un punto e una retta.
`length(`***u***`)` oppure `length(`***s***`)`	Modulo di un vettore o lunghezza di un segmento.
`radius(`***c***`)`	Raggio di una circonferenza.
`major(`***sc***`)`	Il semiasse maggiore se ***sc*** è un'ellisse; il semiasse trasverso se ***sc*** è un'iperbole; il lato retto se ***sc*** è una parabola.
`minor(`***sc***`)`	Il semiasse minore se ***sc*** è un'ellisse; il semiasse non-trasverso se ***sc*** è un'iperbole; zero se ***sc*** è una parabola.

Funzione	Note esplicative
`eccentricity(`***sc***`)`	L'eccentricità di ***sc***.
`arg(`***sc***`, `***A***`)`	Ascissa del punto ***A*** nella rappresentazione parametrica interna (vedi sezione 2.4) della sezione conica ***sc***. Se ***A*** non giace su ***sc*** allora la funzione restituisce l'ascissa di una proiezione del punto sulla conica: la direzione della proiezione è quella dell'asse se ***sc*** è una parabola, quella del centro se ***sc*** è un'ellisse, quella dell'asse trasverso ***sc*** è un'iperbole.
`height(`***A***`, `***B***`, `***C***`)`	Distanza fra il punto ***B*** e la retta ***AC***.

Ci sono inoltre alcune funzioni (tabella 2.2) che restituiscono la misura di un angolo.[2] Gli angoli vanno intesi in ***senso generalizzato***, pertanto '**`angle()`**' restituisce valori compresi fra -180° (escluso) e 180° (compreso).

Tabella 2.2. Eukleides: funzioni che restituiscono misure di angoli.

Funzione	Note esplicative
`angle(`***u***`)` oppure `angle(`***l***`)` oppure `angle(`***s***`)`	Argomento dell'oggetto (rispetto alla rappresentazione interna).
`angle(`***sc***`)`	Argomento dell'asse principale della conica ***sc***.
`angle(`***u***`, `***v***`)`	Misura dell'angolo compreso fra i vettori ***u*** e ***v***.
`angle(`***A***`, `***B***`, `***C***`)`	Misura dell'angolo $\angle ABC$. Si tratta dell'angolo definito dalla rotazione antioraria attorno al punto ***B*** che porta il punto ***A*** sulla retta ***BC***, pertanto '**`angle(`*A*`, `*B*`, `*C*`) + angle(`*C*`, `*B*`, `*A*`)`**' vale 0°, tenuto conto del fatto che è necessario riportare le misure degli angoli nell'intervallo fra -180° e 180°.

2.3 Vettori

Il calcolo vettoriale è possibile attraverso i simboli: '**`(`**', '**`)`**', '**`+`**', '**`-`**', '**`*`**' (moltiplicazione per un numero oppure ***prodotto scalare*** di due vettori), '**`/`**' (divisione per un numero).

La tabella 2.3 riepiloga le funzioni che restituiscono un vettore.

Tabella 2.3. Eukleides: funzioni che restituiscono un vettore.

Funzione	Note esplicative
`vector(`***x***`, `***y***`)`	Vettore definito tramite coordinate cartesiane.
`vector(`***x***`, `***a***`)`	Vettore definito tramite coordinate polari.
`vector(`***A***`, `***B***`)`	Vettore definito dal segmento orientato ***AB*** (ossia ***B***-***A***).
`vector(`***l***`)`	Versore (vettore unitario) parallelo a ***l***.

Funzione	Note esplicative
`vector(`***s***`)`	Vettore definito dal segmento ***s***.
`rotation(`***u***`, `***a***`)`	Immagine del vettore ***u*** secondo la rotazione di ampiezza ***a***.

2.4 Punti

La tabella 2.4 riepiloga le funzioni che restituiscono dei punti.

Tabella 2.4. Eukleides: funzioni che restituiscono dei punti.

Funzione	Note esplicative
`point(`***x***`, `***y***`)`	Punto definito mediante coordinate cartesiane.
`point(`***x***`, `***a***`)`	Punto definito mediante coordinate polari.
`point(`***l***`, `***x***`)`	Punto di ascissa ***x*** lungo la retta ***l*** (in Eukleides le rette hanno una rappresentazione interna e quindi un'orientazione implicita, v. 2.5).
`point(`***s***`, `***x***`)`	Punto di ascissa ***x*** lungo la retta contenente il segmento ***s*** (come le rette anche i segmenti hanno una rappresentazione interna e un'orientazione implicita; l'origine delle ascisse è il primo estremo di ***s***).
`point(`***c***`, `***a***`)`	Punto su ***c*** di argomento ***a*** (secondo l'usuale rappresentazione polare di ***c***).
`point(`***sc***`, `***x***`)`	Punto su ***sc***, ove ***x*** è l'ascissa del punto sulla curva secondo la ***rappresentazione parametrica interna*** di ***sc***: per le parabole la rappresentazione si basa sulla funzione quadratica, per le ellissi sulle funzioni seno e coseno, per le iperboli sulle funzioni secante e tangente.
`barycenter(`***A***`[, `***x***`], `***B***`[, `***y***`])`	Baricentro del segmento ***AB*** con pesi degli estremi rispettivamente ***x*** e ***y*** (valori predefiniti: 1 e 1). Vale una sintassi analoga per tre oppure quattro punti.
`intersection(`***l***`, `***l′***`)`	Intersezione delle rette ***l*** e ***l′***.
`abscissa(`***l***`, `***x***`)`	Punto di ascissa (prima coordinata cartesiana) ***x*** che appartiene alla retta ***l***.
`ordinate(`***l***`, `***y***`)`	Punto di ordinata (seconda coordinata cartesiana) ***y*** che appartiene alla retta ***l***.
`midpoint(`***s***`)`	Punto medio del segmento ***s***.
`begin(`***s***`)`	Primo estremo del segmento ***s***.
`end(`***s***`)`	Secondo estremo del segmento ***s***.
`center(`***c***`)`	Centro della circonferenza ***c***.
`center(`***sc***`)`	Centro della sezione conica ***sc***. Se ***sc*** è una parabola allora restituisce il vertice.
`orthocenter(`***A***`, `***B***`, `***C***`)`	Ortocentro del triangolo ***ABC***.
`translation(`***A***`, `***u***`)`	Immagine del punto ***A*** secondo la traslazione definita dal vettore ***u***.

Funzione	Note esplicative
`reflection(`***A***`, `***l***`)`	Immagine del punto ***A*** secondo la simmetri assiale di asse ***l***.
`rotation(`***A***`, `***B***`[, `***a***`]`)	Immagine del punto A secondo la rotazione di centro ***B*** e ampiezza ***a*** (valore predefinito 180°).
`homothecy(`***A***`, `***B***`, `***x***`)`	Immagine del punto A secondo l'omotetia di centro ***B*** e rapporto ***x***.
`projection(`***A***`, `***l***`[, `***l′***`]`)	Proiezione del punto ***A*** sulla retta ***l*** lungo la direzione definita da ***l′*** (direzione predefinita ortogonale a ***l***).

2.5 Rette

Eukleides rappresenta internamente le linee rette mediante un'ascissa, un'ordinata e un angolo che ne indica l'inclinazione secondo un sistema di riferimento polare (***argomento***), sicché le rette possiedono un'orientazione implicita. La tabella 2.5 riepiloga tutte le funzioni che restituiscono una retta.

Tabella 2.5. Eukleides: funzioni che restituiscono una retta.

Funzione	Note esplicative
`line(`***A***`, `***B***`)`	Retta passante per ***A*** e ***B***.
`line(`***A***`, `***u***`)`	Retta passante per ***A*** e diretta secondo il vettore ***u***.
`line(`***s***`)`	Retta contenente il segmento ***s***.
`line(`***A***`, `***a***`)`	Retta passante per ***A*** e argomento ***a***.
`line(`***c***`, `***a***`)`	Retta tangente a ***c*** nel punto di argomento ***a*** (con polo coincidente con il centro di ***c***).
`line(`***sc***`, `***x***`)`	Retta tangente a ***sc*** nel punto di ascissa ***x*** (rispetto alla rappresentazione parametrica interna di ***sc***: per le parabole la rappresentazione si basa sulla funzione quadratica, per le ellissi sulle funzioni seno e coseno, per le iperboli sulle funzioni secante e tangente).
`parallel(`***l***`, `***A***`)` oppure `parallel(`***s***`, `***A***`)`	Retta passante per ***A*** e parallela a ***l*** (o a ***s***).
`perpendicular(`***l***`, `***A***`)` oppure `perpendicular(`***s***`, `***A***`)`	Retta passante per ***A*** e ortogonale a ***l*** (o a ***s***).
`bisector(`***s***`)`	Asse del segmento ***s***.
`bisector(`***A***`, `***B***`, `***C***`)`	Bisettrice dell'angolo $\angle \boldsymbol{ABC}$.
`bisector(`***l***`, `***l′***`)`	Bisettrice dell'angolo acuto formato da ***l*** e ***l′***.
`altitude(`***A***`, `***B***`, `***C***`)`	Altezza del triangolo ***ABC*** passante per ***A***.
`median(`***A***`, `***B***`, `***C***`)`	Mediana del triangolo ***ABC*** passante per ***A***.

Funzione	Note esplicative
`translation(`***l***`,` ***u***`)`	Immagine di ***l*** secondo la traslazione definita dal vettore ***u***.
`reflection(`***l***`,` ***l′***`)`	Immagine di ***l*** secondo la simmetria assiale di asse ***l′***.
`rotation(`***l***`,` ***A***`[,` ***a***`])`	Immagine di ***l*** secondo la rotazione di centro ***A*** e ampiezza ***a*** (valore predefinito 180°).
`homothecy(`***l***`,` ***A***`,` ***x***`)`	Immagine di ***l*** secondo l'omotetia di centro ***A*** e rapporto ***x***.

2.6 Segmenti

Eukleides rappresenta internamente i segmenti mediante i suoi estremi, pertanto i segmenti possiedono un'orientazione implicita. La tabella 2.6 riepiloga le funzioni che restituiscono un segmento.

Tabella 2.6. Eukleides: funzioni che restituiscono un segmento.

Funzione	Note esplicative
`segment(`***A***`,` ***B***`)`	Segmento di primo estremo ***A*** e secondo estremo ***B***.
`segment(`***A***`,` ***u***`)`	Segmento passante per ***A*** ed equipollente a ***u***.
`segment(`***A***`,` ***x***`,` ***a***`)`	Segmento con primo estremo ***A***, lunghezza ***x*** e argomento ***a***.
`segment(`***c***`,` ***a***`)`	Segmento con primo estremo nel centro di ***c*** e secondo estremo nel punto di ***c*** di argomento ***a*** (polo nel centro di ***c***).
`translation(`***s***`,` ***u***`)`	Immagine di ***s*** secondo la traslazione definita da ***u***.
`reflection(`***s***`,` ***l***`)`	Immagine di ***s*** secondo la simmetria assiale di asse ***l***.
`rotation(`***s***`,` ***A***`[,` ***a***`])`	Immagine di ***s*** secondo la rotazione di centro ***A*** e ampiezza ***a*** (valore predefinito 180°).
`homothecy(`***s***`,` ***A***`,` ***x***`)`	Immagine di ***s*** secondo l'omotetia di centro ***A*** e rapporto ***x***.

2.7 Circonferenze

La tabella 2.7 riepiloga le funzioni che restituiscono una circonferenza.

Tabella 2.7. Eukleides: funzioni che restituiscono una circonferenza.

Funzione	Note esplicative
`circle(`***A***`,` ***B***`)`	Circonferenza di diametro ***AB***.
`circle(`***A***`,` ***B***`,` ***C***`)`	Circonferenza circoscritta al triangolo ***ABC***.
`circle(`***A***`,` ***x***`)`	Circonferenza di centro ***A*** e raggio ***x***.
`incircle(`***A***`,` ***B***`,` ***C***`)`	Circonferenza inscritta nel triangolo ***ABC***.

Funzione	Note esplicative
`translation(`***c***`,` ***u***`)`	Immagine di ***c*** secondo la traslazione definita da ***u***.
`reflection(`***c***`,` ***l***`)`	Immagine di ***c*** secondo la simmetria assiale di asse ***l***.
`rotation(`***c***`,` ***A***`[,` ***a***`])`	Immagine di ***c*** secondo la rotazione di centro ***A*** e ampiezza ***a*** (valore predefinito 180°).
`homothecy(`***c***`,` ***A***`,` ***x***`)`	Immagine di ***c*** secondo l'omotetia di centro ***A*** e rapporto ***x***.

Ci sono inoltre i due assegnamenti multipli descritti nella tabella 2.8.

Tabella 2.8. Eukleides: assegnamenti che restituiscono le intersezioni relative a circonferenze e rette.

Assegnamento	Note esplicative
A ***B*** `intersection(`***l***`,` ***c***`)`	Se i due oggetti sono tangenti allora ***A*** e ***B*** denoteranno il medesimo punto.
A ***B*** `intersection(`***c***`,` ***c′***`)`	Se i due oggetti sono tangenti allora ***A*** e ***B*** denoteranno il medesimo punto.

2.8 Sezioni coniche

La tabella 2.9 riepiloga le funzioni che restituiscono delle sezioni coniche.

Tabella 2.9. Eukleides: funzioni che restituiscono delle sezioni coniche.

Funzione	Note esplicative
`conic(`***A***`,` ***l***`,` ***x***`)`	Sezione conica con fuoco ***A***, direttrice ***l***, eccentricità ***x***.
`conic(`***A***`,` ***B***`,` ***x***`)`	Sezione conica con fuochi ***A*** e ***B***. Il valore ***x*** è la semidistanza fra i vertici (ossia l'asse maggiore per le ellissi, l'asse trasverso per le iperboli).
`parabola(`***A***`,` ***l***`)`	Parabola con fuoco ***A*** e direttrice ***l***.
`parabola(`***A***`,` ***x***`,` ***a***`)`	Parabola con vertice ***A***, con ***x*** pari al lato retto e ***a*** pari all'argomento dell'asse.
`ellipse(`***A***`,` ***x***`,` ***y***`,` ***a***`)`	Ellisse con centro ***A***, semiassi maggiore e minore ***x*** e ***y***, argomento dell'asse principale pari a ***a***.
`hyperbola(`***A***`,` ***x***`,` ***y***`,` ***a***`)`	Iperbole con centro ***A***, semiassi reale e immaginario ***x*** e ***y***, argomento dell'asse principale pari a ***a***.
`translation(`***sc***`,` ***u***`)`	Immagine di ***sc*** secondo la traslazione definita da ***u***.
`reflection(`***sc***`,` ***l***`)`	Immagine di ***sc*** secondo la simmetria assiale di asse ***l***.
`rotation(`***sc***`,` ***A***`[,` ***a***`])`	Immagine di ***sc*** secondo la rotazione di centro ***A*** e ampiezza ***a*** (valore predefinito 180°).
`homothecy(`***sc***`,` ***A***`,` ***x***`)`	Immagine di ***sc*** secondo l'omotetia di centro ***A*** e rapporto ***x***.

La tabella 2.10 riepiloga alcuni assegamenti multipli che coinvolgono sezioni coniche.

Tabella 2.10. Eukleides: assegnamenti multipli che coinvolgono sezioni coniche.

Assegnamento	Note esplicative
A B `foci(`***sc***`)`	Fuochi della sezione conica ***sc***.
A B `vertices(`***sc***`)`	Vertici della sezione conica ***sc***.
A B `intersection(`***l***`, `***sc***`)`	Punti di intersezione fra ***l*** e ***sc***.

2.9 Triangoli

Un ***assegnamento di un triangolo*** è costituito da una lista di tre nomi di variabile seguita da una delle parole chiave '**`triangle`**', '**`right`**', '**`isosceles`**' oppure '**`equilateral`**' più alcuni parametri opzionali. Se alla prima variabile è già stato assegnato un punto allora il triangolo viene costruito a partire da tale punto, altrimenti si parte dall'origine. Il paramentro opzionale ***b*** è l'argomento del segmento ***AB*** (valore predefinito: 0°).

Nella tabella 2.11 vengono illustrati i vari modi per definire un triangolo.

Tabella 2.11. Eukleides: i vari modi per definire un triangolo.

Assegnamento	Note esplicative
A B C `triangle[(`***x***`[, `***b***`])]`	Definisce un triangolo scaleno tale che ***AB*** misuri ***x*** (valore predefinito: 6). Il triangolo scaleno è ***ottimale***, ossia è un triangolo acutangolo la cui forma si discosti il più possibile da quella di un triangolo rettangolo o isoscele.
A B C `triangle(`***x***`, `***y***`, `***z***`[, `***b***`])`	Definisce un triangolo scaleno tale che ***AB*** misuri ***x***, ***BC*** misuri ***y*** e ***AC*** misuri ***z***.
A B C `triangle(`***x***`, `***a***`, `***a′***`[, `***b***`])`	Definisce un triangolo scaleno tale che ***AB*** misuri ***x***, $\angle BAC$ misuri ***a*** e $\angle CBA$ misuri a'.
A B C `right[(`***x***`, `***y***`[, `***b***`])]`	Definisce un triangolo rettangolo in ***A*** tale che ***AB*** misuri ***x*** e ***AC*** misuri ***y*** (valori predefiniti: 6 e 4,5 rispettivamente).
A B C `right(`***x***`, `***a***`[, `***b***`])`	Definisce un triangolo rettangolo in ***B*** tale che ***AB*** misuri ***x*** e la misura di $\angle BAC$ sia ***a***.
A B C `isosceles[(`***x***`, `***a***`[, `***b***`])]`	Definisce un triangolo isoscele tale che la base ***AB*** misuri ***x*** e gli angoli alla base misurino ***a*** (valori predefiniti: 6 e 39° rispettivamente).
A B C `isosceles(`***x***`, `***y***`[, `***b***`])`	Definisce un triangolo isoscele tale che la base ***AB*** misuri ***x*** e i lati obliqui misurino ***y***.
A B C `equilateral[(`***x***`[, `***b***`])]`	Definisce un triangolo equilatero di lato ***x*** (valore predefinito: 6).

2.10 Poligoni

Un ***assegnamento di un quadrilatero*** è costituito da una lista di quattro nomi di variabile seguita da una delle parole chiave '**`parallelogram`**', '**`rectangle`**' oppure '**`square`**' più alcuni parametri opzionali. Se alla prima variabile è di già stato assegnato un punto allora il quadrilatero viene costruito a partire da tale punto, altrimenti si parte dall'origine. Il paramentro opzionale ***b*** è l'argomento del segmento ***AB*** (valore predefinito: 0°).

Nella tabella 2.12 vengono illustrati i vari modi per definire un quadrilatero.

Tabella 2.12. Eukleides: i vari modi per definire un quadrilatero.

Assegnamento	Note esplicative
`A B C D parallelogram[ (x, y, a[, b]) ]`	Definisce un parallelogrammo tale che ***AB*** misuri ***x***, ***AD*** misuri ***y*** e l'angolo $\angle$***BAD*** misuri ***a*** (valori predefiniti: 5, 4 e 75° rispettivamente).
`A B C D parallelogram(u, v[ , b])`	Definisce un parallelogrammo tale che ***B*-*A*** = ***u*** e ***D*-*A*** = ***v***.
`A B C D rectangle[(x, y[, b ])]`	Definisce un rettangolo tale che ***AB*** misuri ***x*** e ***AD*** misuri ***y*** (valori predefiniti: 6 e 6/φ rispettivamente, ove φ è il ***rapporto aureo***).
`A B C D square[(x[, b])]`	Definisce un quadrato di lato ***x*** (valore predefinito: 4).

Esistono inoltre l'***assegnamento di un pentagono*** o ***di un esagono*** (tabella 2.13).

Tabella 2.13. Eukleides: assegnamenti per pentagoni e per esagoni.

Assegnamento	Note esplicative
`A B C D E pentagon(F, x, a)`	Definisce un pentagono regolare di centro ***F***, raggio ***x*** e tale che l'argomento di ***FA*** sia ***a***.
`A B C D E F pentagon(G, x, a)`	Definisce un esagono regolare di centro ***G***, raggio ***x*** e tale che l'argomento di ***GA*** sia ***a***.

2.11 Assegnamenti interattivi

Un ***assegnamento interattivo*** è fatto così:

```
x interactive(y, z[, x', x'',] str, f)
```

Per '**`eukleides`**' tale assegnamento equivale a '***x*** **`=`** ***y***'; usando '**`xeukleides`**' esso permette di modificare il valore di ***x*** **dalla modalità di presentazione** mediante i tasti freccia. Il valore iniziale di ***x*** è ***y*** e l'incremento è ***z***. I parametri opzionali ***x***′ e ***x***″ stabiliscono il minimo e il massimo per ***x***. La stringa ***str*** **deve** consistere di un singolo carattere alfabetico maiuscolo: per modificare ***x*** si deve premere per prima cosa il tasto alfabetico corrispondente.[3] I valori possibili per ***f*** sono '**`right`**' (nel qual caso ***x*** viene incrementato alla pressione del tasto [*freccia destra*] e decrementato alla pressione del tasto [*freccia sinistra*]) oppure '**`up`**' (nel qual caso ***x*** viene incrementato alla pressione del tasto [*freccia su*] e decrementato alla pressione del tasto [*freccia giù*]).

2.12 Comandi di impostazione

La tabella 2.14 riepiloga comandi di impostazione.

Tabella 2.14. Eukleides: comandi di impostazione.

Comando	Note esplicative
{box | frame}(*x*, *y*, *x′*, *y′*[, *z*])	Con '**eukleides**' i due comandi hanno un effetto simile, ossia impostare il riquadro visibile. Il vertice inferiore sinistro ha coordinate cartesiane (***x***,***y***) e quello superiore destro coordinate (***x′***,***y′***) mentre il parametro opzionale imposta l'unità di misura a ***z*** cm (valore predefinito: 1). Con '**xeukleides**' la figura viene disegnata utilizzando la scala massima che renda il riquadro inscritto nell'area di disegno. Il comando '**box**' imposta il riquadro visibile sicché la figura di solito appare con due strisce grigie ai lati. Il comando '**frame**' imposta il **minimo** riquadro visibile sicché viene utilizzata l'intera area di disegno. Il parametro ***z*** viene ignorato. Il riquadro predefinito è: (-2, -2, 8, 6).
color(***f***)	Imposta il colore a ***f***. I valori ammessi sono '**black**' (predefinito), '**darkgray**', '**gray**', '**lightgray**', '**white**', '**red**', '**green**', '**blue**', '**cyan**', '**magenta**' e '**yellow**'.
color ***str***	Con '**eukleides**' imposta il colore a ***str*** (dev'essere un colore valido per '**pstricks**'). Con '**xeukleides**' non ha effetto.
thickness(*x*)	Con '**eukleides**' imposta lo spessore delle linee corrente a ***x*** pt (lo spessore predefinito è 0,5 pt). Con '**xeukleides**' non ha effetto.
style(***f***)	Imposta lo stile di disegno corrente a ***f***. I valori ammessi sono '**full**', '**dashed**' e '**dotted**'. Il valore predefinito è '**full**'.
tricks ***str***	Con '**eukleides**' aggiunge in testa a ***str*** un carattere '\' e emette il risultato. Deve trattarsi di codice TeX valido. Con '**xeukleides**' non ha effetto.
font ***str***	Con '**xeukleides**' cambia la fonte corrente. La stringa ***str*** deve essere un valido nome XLFD (*X Logical Font Description Conventions*). Con '**eukleides**' non ha effetto.

2.13 Comandi di disegno

La tabella 2.15 riepiloga comandi di disegno.

Tabella 2.15. Eukleides: comandi di disegno.

Comando	Note esplicative
draw(***A***[, ***f***[, ***x***]])	Disegna un punto ***A*** con forma definita da ***f*** e rapporto di scala definito da ***x*** (valore predefinito: 1). I valori ammessi per ***f*** sono '**dot**', '**box**', '**cross**' e '**plus**' (valore predefinito: '**dot**'). Il rapporto di scala non ha effetto con '**xeukleides**'.
draw(***u***, ***A***[, ***f***])	Disegna un vettore ***u*** con origine in ***A***. Specificando ***f*** si può modificare lo stile corrente; i valori ammessi sono: '**full**', '**dashed**' e '**dotted**'.
draw(***l***[, ***f***[, ***f′***]])	Disegna una retta. Specificando ***f*** o ***f′*** si può modificare lo stile corrente: i valori ammessi per ***f*** sono '**full**', '**dashed**' e '**dotted**'; i valori ammessi per ***f′*** sono '**entire**', '**halfline**' e '**backhalfline**' (valore predefinito: '**entire**').

Comando	Note esplicative
`draw(`*s*`[, `*f*`[, `*f′*`]])`	Disegna un segmento. Specificando *f* o *f′* si può modificare lo stile corrente: i valori ammessi per *f* sono '**full**', '**dashed**' e '**dotted**'; i valori ammessi per *f′* sono '**noarrow**', '**arrow**', '**backarrow**' e '**doublearrow**' (valore predefinito: '**noarrow**').
`draw(`*c*`[, `*f*`])`	Disegna una circonferenza. Specificando *f* si può modificare lo stile corrente: i valori ammessi sono '**full**', '**dashed**' e '**dotted**'.
`draw(`*c*`, `*a*`, `*a′*`[, `*f*`[, `*f′*`]])`	Disegna arco della circonferenza *c* compreso fra i punti di coordinate polari angolari *a* e *a′*. Specificando *f* o *f′* si può modificare lo stile corrente: i valori ammessi per *f* sono '**full**', '**dashed**' e '**dotted**'; i valori ammessi per *f′* sono '**noarrow**', '**arrow**', '**backarrow**' e '**doublearrow**' (valore predefinito: '**noarrow**').
`draw(`*sc*`[, `*f*`])`	Disegna una sezione conica *sc*. Specificando *f* si può modificare lo stile corrente: i valori ammessi sono '**full**', '**dashed**' e '**dotted**'.
`draw(`*sc*`, `*x*`, `*y*`[, `*f*`])`	Disegna un arco della sezione conica *sc*. I valori *x* e *y* denotano l'ascissa (relativamente alla rappresentazione parametrica interna di *sc*) degli estremi dell'arco. Specificando *f* si può modificare lo stile corrente: i valori ammessi sono '**full**', '**dashed**' e '**dotted**'.
`draw(`*A*`, `*B*`, `*C*`[, `*f*`])`	Disegna il triangolo *ABC*. Specificando *f* si può modificare lo stile corrente: i valori ammessi sono '**full**', '**dashed**' e '**dotted**'.
`draw(`*A*`, `*B*`, `*C*`, `*D*`[, `*f*`])`	Disegna il quadrilatero *ABCD*. Specificando *f* si può modificare lo stile corrente: i valori ammessi sono '**full**', '**dashed**' e '**dotted**'.
`draw(`*A*`, `*B*`, `*C*`, `*D*`, `*E*`[, `*f*`])`	Disegna il pentagono *ABCDE*. Specificando *f* si può modificare lo stile corrente: i valori ammessi sono '**full**', '**dashed**' e '**dotted**'.
`draw(`*A*`, `*B*`, `*C*`, `*D*`, `*E*`, `*F*`[, `*f*`])`	Disegna l'esagono *ABCDEF*. Specificando *f* si può modificare lo stile corrente: i valori ammessi sono '**full**', '**dashed**' e '**dotted**'.
`draw(`*str*`, `*A*`,[ `*x*`,] `*a*`)`	Scrive il testo contenuto in *str* a una distanza *x* (valore predefinito: 0,3 cm) e con argomento *a* rispetto al polo *A*.
`draw(`*str*`, `*s*`,[ `*x*`,] `*a*`)`	Scrive il testo contenuto in *str* a una distanza *x* (valore predefinito: 0,3 cm) e con argomento *a* rispetto al punto medio del segmento *s*.
`draw(`*x*`,[ `*str*`,] `*A*`,[ `*y*`,] `*a*`)`	Scrive il valore di *x*, eventualmente secondo il formato definito da *str* (sintassi del linguaggio C), a una distanza *y* (valore predefinito 0,3 cm) e con argomento *a* rispetto al polo *A*.
`draw(`*x*`,[ `*str*`,] `*s*`,[ `*y*`,] `*a*`)`	Scrive il valore di *x*, eventualmente secondo il formato definito da *str* (sintassi del linguaggio C), a una distanza *y* (valore predefinito 0,3 cm) e con argomento *a* rispetto al punto medio del segmento *s*.
`draw(`*x*`, `*x′*`, `*str*`, `*A*`,[ `*y*`,] `*a*`)`	Scrive i valori di *x* e *x′*, eventualmente secondo il formato definito da *str* (sintassi del linguaggio C), a una distanza *y* (valore predefinito 0,3 cm) e con argomento *a* rispetto al polo *A*.
`draw(`*x*`, `*x′*`, `*str*`, `*s*`,[ `*y*`,] `*a*`)`	Scrive i valori di *x* e *x′*, eventualmente secondo il formato definito da *str* (sintassi del linguaggio C), a una distanza *y* (valore predefinito 0,3 cm) e con argomento *a* rispetto al punto medio del segmento *s*.

Comando	Note esplicative
`mark(`*s*`[, `*f*`[, `*x*`]])`	Marca il segmento ***s*** con un simbolo definito da ***f*** usando un rapporto di scala ***x*** (valore predefinito: 1). I valori ammessi per ***f*** sono: '**`simple`**', '**`double`**', '**`triple`**' e '**`cross`**'.
`mark(`*A*`, `*B*`, `*C*`[, `*f*`[, `*x*`]])`	Marca l'angolo $\angle ABC$ con un simbolo definito da ***f*** usando un rapporto di scala ***x*** (valore predefinito: 1). I valori ammessi per ***f*** sono: '**`simple`**', '**`double`**', '**`triple`**', '**`dashed`**' (quest'ultimo produce una marcatore curvo con un tratto trasversale), '**`right`**', '**`dotted`**' (quest'ultimo produce una marcatore di angolo retto con un punto al centro), '**`arrow`**' e '**`backarrow`**' (valore predefinito: '**`simple`**').

2.14 Comandi di tracciamento

Un ***comando di tracciamento*** permette di disegnare un luogo geometrico al variare di un parametro. Ecco la sintassi:

```
trace(x, x′, x″[, f])
```

Il comando deve essere seguito da almeno un'espressione che restituisca un punto, inclusa fra parentesi graffe; l'espressione deve dipendere dalla variabile x. I paramteri x' e x'' denotano rispettivamente il minimo e il massimo per x. L'espressione deve essere definita in tutto l'intevallo, altrimenti si possono ottenere risultati incoerenti o a volte (con '**`eukleides`**') dei messaggi di errore. All'interno delle parentesi graffe, al fine di eseguire calcoli intermedi, è possibile far precedere qualsiasi successione di comandi all'espressione. Specificando ***f*** si può modificare lo stile corrente: i valori ammessi sono '**`full`**', '**`dashed`**' e '**`dotted`**'.

2.15 «pstricks»

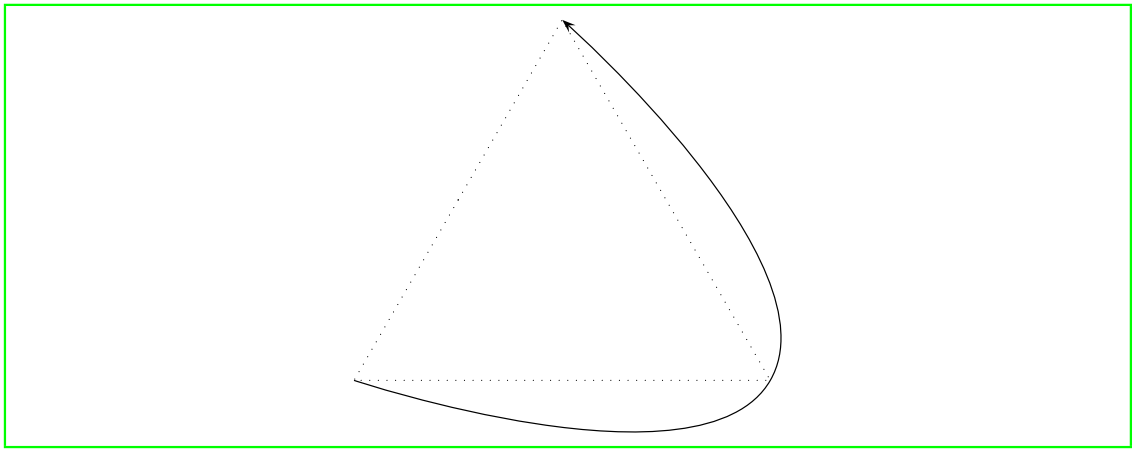
Il programma '**`eukleides`**' permette di includere dei comandi del pacchetto di macro '**`pstricks`**' utilizzando parametri gestiti da '**`eukleides`**' stesso. L'utilizzo probabilmente più opportuno di questa funzionalità è quello di disegnare degli oggetti '**`pstricks`**' con coordinate generate da una costruzione in linguaggio Eukleides.

La sintassi è ragionevolmente simile a quella che si utilizza nell'inserire codice '**`pstricks`**' in un documento TeX o LaTeX. L'invocazione di una macro di '**`pstricks`**' inizia con il simbolo '**`\`**' e prosegue con il nome della macro stessa (ad esempio: '**`pscircle`**'). Può esserci anche una lista di opzioni per la macro, inclusa fra parentesi quadre, e anche una lista di parametri fra parentesi tonde. Un semplice esempio è presentato nel listato 2.16 e nella corrispondente figura 2.17; nel seguito varranno date spiegazioni più dettagliate.

Listato 2.16. Eukleides: esempio elementare con '**`pstricks`**'.

```
A B C equilateral
draw(A, B, C, dotted)
\pscurve[arrows="->"](A, B, C)
```

Figura 2.17.



2.15.1 Opzioni

Le eventuali opzioni devono essere specificate nella forma

opzione=valore

dove ***opzione*** è un'opzione prevista da '**pstricks**' e ***valore*** può essere una stringa fra doppi apici (ad esempio: '**arrows="->"**') oppure un espressione numerica di Eukleides (ad esempio: '**radius=distance(A,B)**'). I doppi apici verranno eliminati dal programma prima dell'emissione del codice '**pstricks**'.

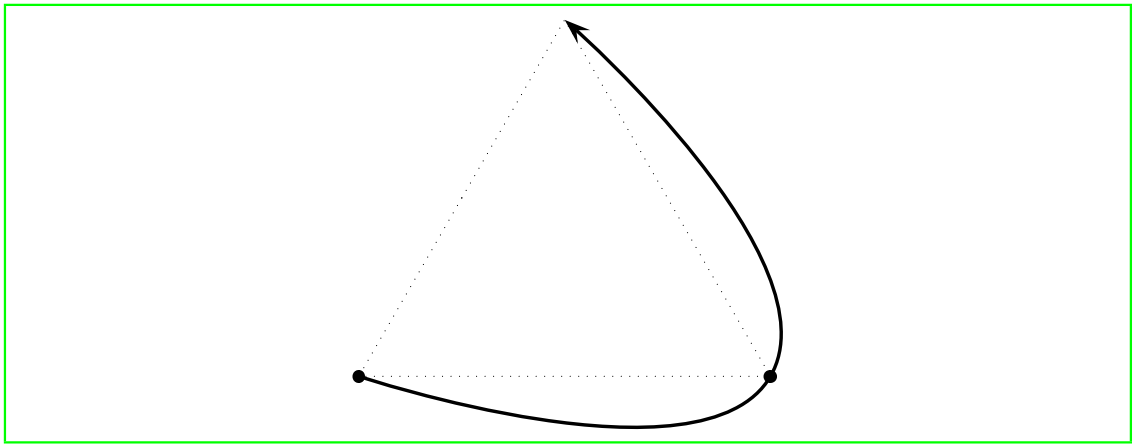
Tutte le opzioni di '**pstricks**' che siano documentate dovrebbero funzionare senza il bisogno di includerle fra doppi apici, ma in caso di problemi è possibile farlo (ad esempio: '**"parametroStrano"=3.145**').

A titolo di esempio, si considerino il listato 2.18 e la corrispondente figura 2.19 (che modificano lievemente il listato 2.16 e la figura 2.17, rispettivamente).

Listato 2.18. Eukleides: altro esempio elementare con '**pstricks**'.

```
A B C equilateral
draw(A, B, C, dotted)
\pscurve[showpoints="true", linewidth=.05, arrows="->"](A, B, C)
```

Figura 2.19.



2.15.2 Parametri

I parametri da passare alle macro di '**pstricks**' devono essere specificati come una lista, separata da virgole, di espressioni che restituiscano punti oppure valori numerici o di stringhe fra doppi apici. I punti vengono trasformati nella forma '**(*x*,*y*)**' e le stringhe fra doppi apici e le espressioni numeriche vengono inserite all'interno delle parentesi graffe; ad esempio, il comando '**\psline("<<->>", A, B)**' viene tradotto nel codice '**pstricks**' seguente:

```
\psline{<<->>}(0.0000,0.0000)(6.0000,0.0000)
```

Le indicazioni qui fornite non pretendono minimamente di essere una spiegazione - nemmeno superficiale - del funzionamento del pacchetto di macro '**pstricks**'. Per maggiori informazioni si veda la sezione 6.
Inoltre, si tenga presente che non viene effettuato nessun controllo sul codice '**pstricks**' che viene generato da '**eukleides**', il che significa che è possibile generare codice TeX sintatticamente scorretto se si utilizza questa caratteristica di '**eukleides**'.
Infine si noti che al momento della presente stesura '**xeukleides**' ignora completamente le macro '**pstricks**'.

[1] Usando '**xeukleides**', gli eventuali simboli '**$**' presenti nelle stringhe vengono ignorati, allo scopo di migliorare la leggibilità della presentazione; per '**eukleides**' invece tali simboli delimitano del codice TeX che cerrà compilato e inserito nella figura.

[2] in gradi

[3] All'inizio lo stato predefinito è "A".

Limitazioni e punti di forza

Le limitazioni principali di Eukleides sono probabilmente:

1. il basso livello di interattività del programma frontale '**`xeukleides`**', e
2. la mancanza di strutture sintattiche più evolute (cicli, sottoprogrammi, ecc...) nel linguaggio vero e proprio.

Per quanto riguarda il primo punto, la questione è complessa e si potrebbe inquadrare più nell'ambito delle scelte metodologiche che in quello dei compromessi tecnologici. In effetti esistono già diversi programmi che permettono di «fare geometria dinamica», nel senso che permettono all'utente di costruire e modificare gli oggetti geometrici essenzialmente «a colpi di mouse»: non è questa la sede per aprire una discussione sulla presunta opportunità didattica di strumenti di questo tipo (basti notare che al riguardo le diverse opinioni dei docenti di matematica sono largamente divergenti); fatto sta che la scelta dell'autore di Eukleides (egli stesso insegnante liceale di matematica in Francia) è stata diversa:

> As a mathematics teacher in a French high school, I have to compose a rather large number of documents for my students, containing both text and formulas. In my point of view, LaTeX is the best tool in such a situation, combining efficiency and high quality. Very often, these documents should be illustrated with geometric figures. I first used the excellent '**`pstrick`**' package to draw them. I didn't want to use WYSIWYG software instead, because I wanted to keep following TeX's philosophy, that is: What You Mean Is What You Get. Unfortunately, '**`pstrick`**' isn't designed for geometry at all and is rather inappropriate in many situations.
>
> One night, I wanted to draw a triangle with an inscribed circle, so I had to compute by hand the coordinates of the center and the radius of this circle, which is quite boring. During these calculations, I realized that they could easily be done by a computer, and that gave me the idea to create Eukleides, a geometry drawing language. *My goal was to make it as close as possible to what mathematics teachers would say to describe geometric figures.*[1]
>
> [...]

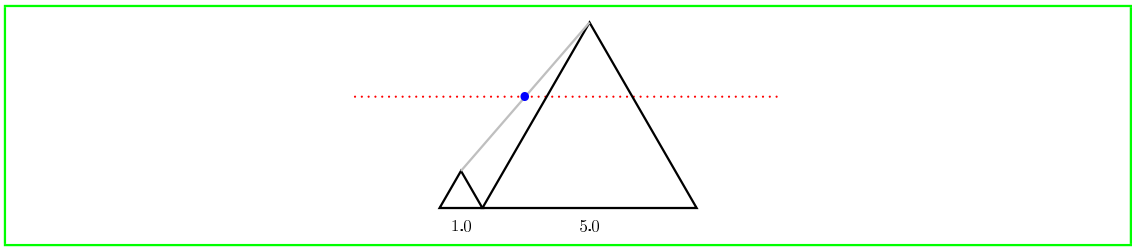
Sembra quindi che lo scopo principale di Eukleides sia quello di (facilitare la e) invitare alla **descrizione** verbale delle figure geometriche, piuttosto che coltivare un'attività semi-ludica di modifica interattiva delle figure stesse (cosa comunque praticamente possibile, anche se «senza mouse», grazie agli assegnamenti interattivi, v. sezione 2.11). Peraltro, avere a che fare con un software *orientato alla descrizione* offre anche tutta una serie di indiscutibili vantaggi (come si cercherà di dimostrare nel seguito).

Riguardo al secondo punto (fermo restando che le future versioni di Eukleides conterranno dei decisi miglioramenti in questo senso), proprio grazie alla natura descrittivo-testuale del software in oggetto è possibile «estendere» le funzionalità del linguaggio utilizzando la flessibilità offerta dai diversi linguaggi di programmazione. Si consideri ad esempio il seguente enunciato: «Sia dato un segmento e un punto su di esso; si costruiscano - sulle due parti del segmento così definite - due triangoli equilateri dalla stessa parte del piano rispetto al segmento, e si consideri il segmento avente per estremi i vertici non giacenti sul segmento dato; il punto medio di tale segmento descrive - al variare del punto dato - un segmento su una retta parallela al segmento dato». Per tentare di fornire una «dimostrazione dinamica» dell'enunciato, si parta dal caso particolare fornito dal listato 3.1 e dalla figura 3.2.

Listato 3.1.

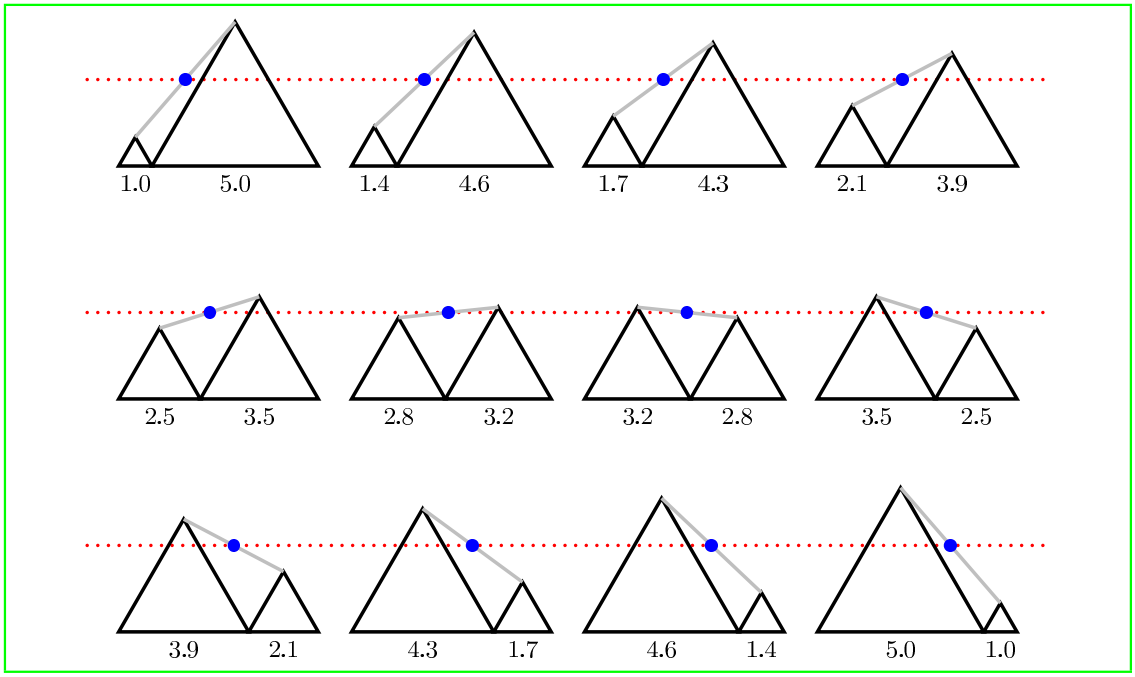
```
thickness(3)
H = point(0,3*3^(1/2)/2)
color(red)
style(dotted)
% la retta che dovrebbe contenere il luogo:
draw(line(H,0:))
color(black)
style(full)
A = point(0,0)
AM = 1
MB = 6-AM
A M I equilateral(AM)
M B J equilateral(MB)
draw(A,M,I) ; draw(M,B,J)
color(lightgray)
draw(segment(I,J))
color(black)
draw(AM,"%.1f",segment(A,M),-90:)
draw(MB,"%.1f",segment(M,B),-90:)
color(blue)
% il punto che descrive il luogo:
draw(barycenter(I,J))
```

Figura 3.2. Eukleides: un problema di luoghi geometrici (un caso particolare).



Naturalmente, una «dimostrazione dinamica» richiede di poter disegnare un gran numero di casi particolari simili a quello illustrato, mettendo così in evidenza la limitazione di Eukleides legata al non poter definire cicli enumerativi. Tuttavia, utilizzando dei linguaggi di programmazione esterni, è abbastanza semplice arrivare a figure come la 3.3.

Figura 3.3. Eukleides: una «dimostrazione dinamica».



Per realizzare tale figura, si è utilizzato il programma '**eukloop.sh**' (il file 'eukloop_sh' contenente il programma dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro), a partire dai file '2triloc.eukt' e '2triseq.eukt'. Ecco come si procede:

$ echo "frame(-1,-16,28,6,0.5)" > 2tri.euk ; echo "thickness(3)" >> 2tri.euk [*Invio*]

$ eukloop.sh [*Invio*]

```
create symbolic link 'eukloop1.sh' to 'eukloop.sh'
create symbolic link 'eukloop2.sh' to 'eukloop.sh'
```

$ cat 2triloc.eukt [*Invio*]

```
y = <COUNTER1>
H = point(0,-7*y+3*3^(1/2)/2)
color(red)
style(dotted)
draw(line(H,0:))
color(black)
style(full)
```

$ cat 2triloc.eukt | eukloop1.sh 0 1 2 >> 2tri.euk [*Invio*]

$ cat 2triseq.eukt [*Invio*]

```
x = <COUNTER2>
y = <COUNTER1>
A = point(7*x,-7*y)
AM = 4/11*x+16/11*y+1
MB = 6-AM
```

```
A M I equilateral(AM)
M B J equilateral(MB)
draw(A,M,I) ; draw(M,B,J)
color(lightgray)
draw(segment(I,J))
color(black)
draw(AM,"%.1f",segment(A,M),-90:)
draw(MB,"%.1f",segment(M,B),-90:)
color(blue)
draw(barycenter(I,J))
color(black)
```

```
$ cat 2triseq.eukt | eukloop2.sh 0 1 2 0 1 3 >> 2tri.euk [ Invio ]

$
```

Il file '`2tri.euk`' dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro): esso corrisponde esattamente alla figura 3.3.

Il funzionamento del programma '**`eukloop.sh`**' è molto semplice:[2] invocandolo con il suo nome esso crea due collegamenti simbolici a se stesso nella directory corrente, che verranno poi utilizzati nell'invocazione effettiva, ciascuno di essi corrispondendo a una differente modalità di utilizzo; il primo collegamento simbolico realizza una struttura ciclica enumerativa semplice, il secondo una composta da due cicli annidati; in pratica il programma converte un file in cui sia presente del codice Eukleides assieme alle metavariabili <***COUNTER1***> e <***COUNTER2***> in un file Eukleides ottenuto ripetendo il codice Eukleides e sostituendo alle metavariabili dei valori specifici ottenuti ciclicamente in conformità agli argomenti passati al programma; nel caso di '**`eukloop1.sh`**' i tre argomenti passati corrispondono rispettivamente al valore iniziale, all'incremento e al valore finale (e analogamente nel caso di '**`eukloop2.sh`**', solo che in questo caso bisogna passare tre valori per ciascuna metavariabile, per un totale di sei argomenti); nel caso del ciclo doppio, il ciclo esterno corrisponde alla metavariabile <***COUNTER1***>, quello interno alla metavariabile <***COUNTER2***>.

[1] enfasi mia

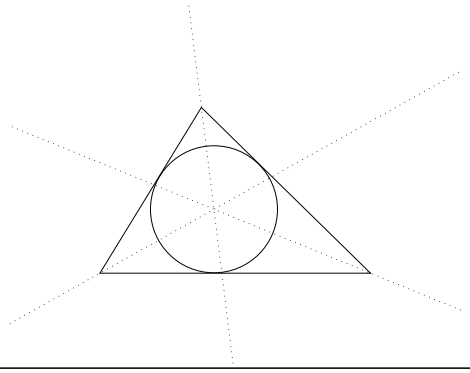
[2] ... per non dire rudimentale: si invita il lettore volenteroso ad estenderne le funzionalità, magari utilizzando un altro linguaggio di programmazione.

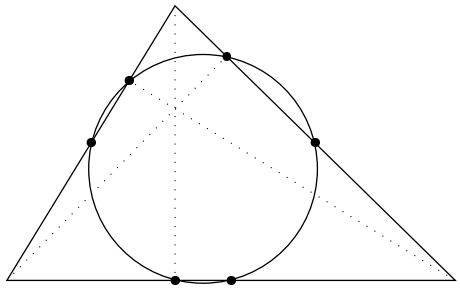
Esempi

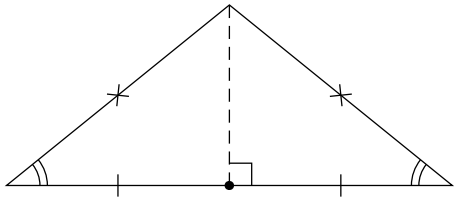
Gli esempi presentati nella tabella 4.1 illustrano le caratteristiche fondamentali di Eukleides; nella sezione 5 vengono presentati degli esempi più avanzati.

Tabella 4.1. Eukleides: esempi elementari.

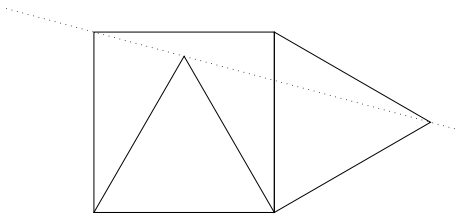
Codice Eukleides	Figura risultante

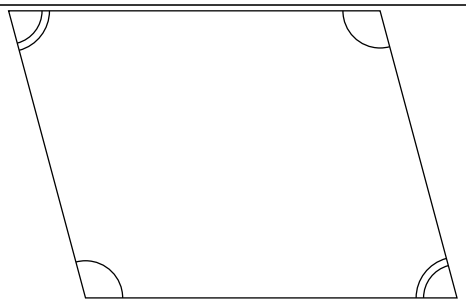
`% Circonferenza inscritta e bisettrici`
`A B C triangle`
`draw(A, B, C)`
`draw(incircle(A, B, C))`
`draw(bisector(B, A, C), dotted)`
`draw(bisector(A, B, C), dotted)`
`draw(bisector(B, C, A), dotted)`
	

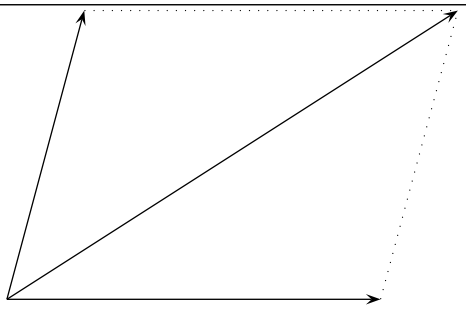
`% Circolo di Feuerbach`
`A B C triangle`
`a = projection(A, line(B, C))`
`b = projection(B, line(A, C))`
`c = projection(C, line(A, B))`
`draw(A, B, C)`
`draw(a) ; draw(b) ; draw(c)`
`draw(segment(A, a), dotted)`
`draw(segment(B, b), dotted)`
`draw(segment(C, c), dotted)`
`draw(barycenter(A, B))`
`draw(barycenter(B, C))`
`draw(barycenter(C, A))`
`draw(circle(a, b, c))`
	

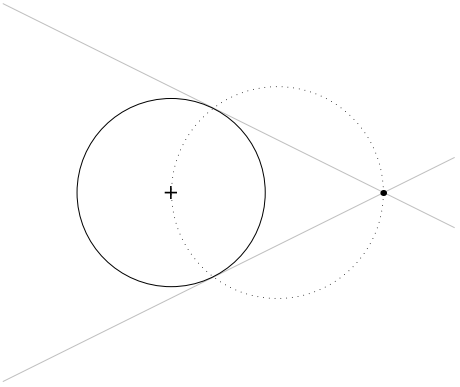
`% Triangolo isoscele`
`A B C isosceles`
`H = projection(C, line(A, B))`
`draw(A, B, C)`
`draw(H)`
`draw(segment(C, H), dashed)`
`mark(B, H, C, right)`
`mark(segment(A, H))`
`mark(segment(B, H))`
`mark(segment(A, C), cross)`
`mark(segment(C, B), cross)`
`mark(B, A, C, double)`
`mark(C, B, A, double)`
	

Codice Eukleides	Figura risultante

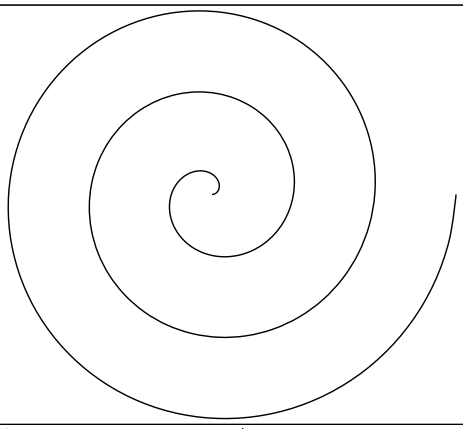
`% Punti allineati`
`A B C D square`
`A B E equilateral(4)`
`B F G equilateral(4, 30:)`
`draw(A, B, C, D)`
`draw(A, B, E)`
`draw(B, F, G)`
`draw(line(E, F), dotted)`
	

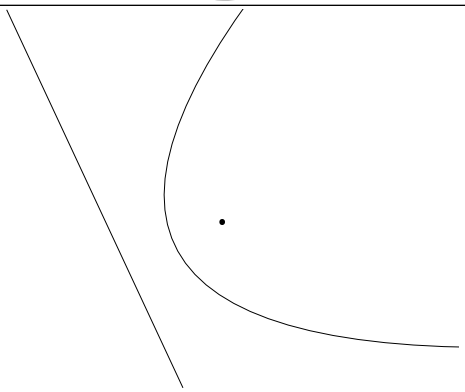
`% Angoli di un parallelogrammo`
`A B C D parallelogram(5, 4, 105:)`
`draw(A, B, C, D)`
`mark(B, A, D)`
`mark(D, C, B)`
`mark(C, B, A, double)`
`mark(A, D, C, double)`
	

`% Somma vettoriale`
`A B C D parallelogram`
`draw(segment(A, B), full, arrow)`
`draw(segment(A, C), full, arrow)`
`draw(segment(A, D), full, arrow)`
`draw(segment(B, C), dotted)`
`draw(segment(D, C), dotted)`
	

`% Tangenti a una circonferenza`
`O = point(2, 2)`
`C = circle(O, 2)`
`A = point(6.5, 2)`
`c = circle(O, A)`
`I J intersection(C, c)`
`color(lightgray)`
`draw(line(A, I))`
`draw(line(A, J))`
`color(black)`
`draw(O, plus)`
`draw(A)`
`draw(C)`
`draw(c, dotted)`
	

Codice Eukleides	Figura risultante

`% Una spirale`
`frame(-5, -4, 5, 4)`
`trace(t, 0, 3*360)`
`{ point(t/360, t:) }`
	

`% Parabola con fuoco e direttrice`
`F = point(3, 1.5)`
`D = line(point(1, 0.5), -65:)`
`C = parabola(F, D)`
`draw(F)`
`draw(D)`
`draw(C)`
	

Altri esempi

Le figure presentate in questa sezione derivano dagli esempi inclusi nella distribuzione standard del pacchetto Debian.

Figura 5.1. Eukleides: metodo attribuito a Abdul al Wafa per la costruzione di un triangolo equilatero inscritto in un quadrato (il file `abdul_al_wafa_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

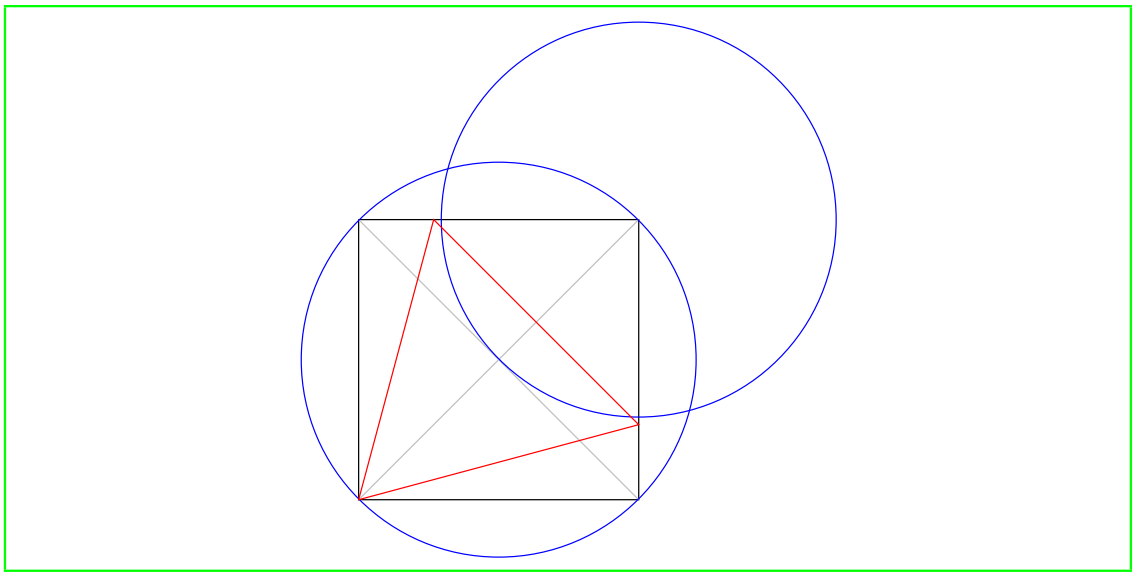


Figura 5.2. Eukleides: versiera di Agnesi (il file `agnesi_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

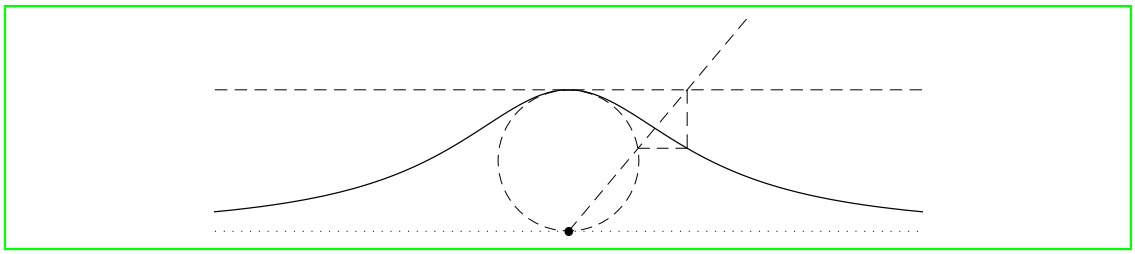


Figura 5.3. Eukleides: illustrazione per un esercizio sulla somma degli angoli α e β (il file `angles_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

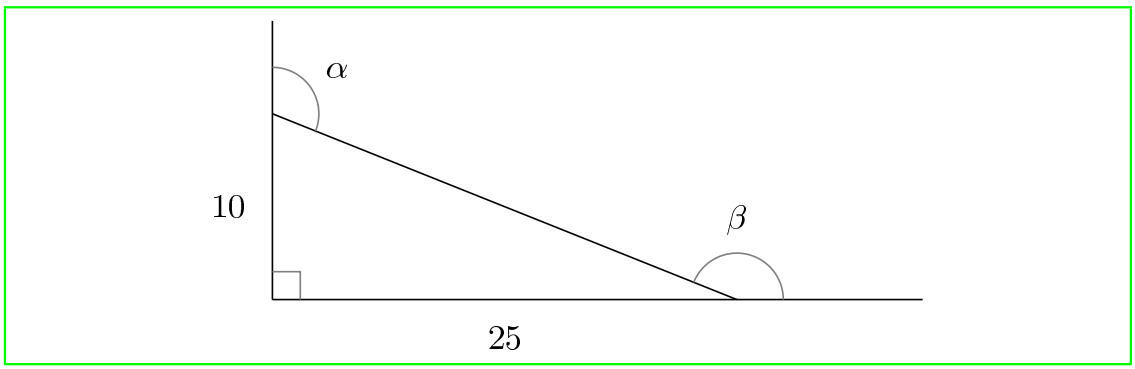


Figura 5.4. Eukleides: bisettrici in un parallelogrammo (il file `bisectors_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

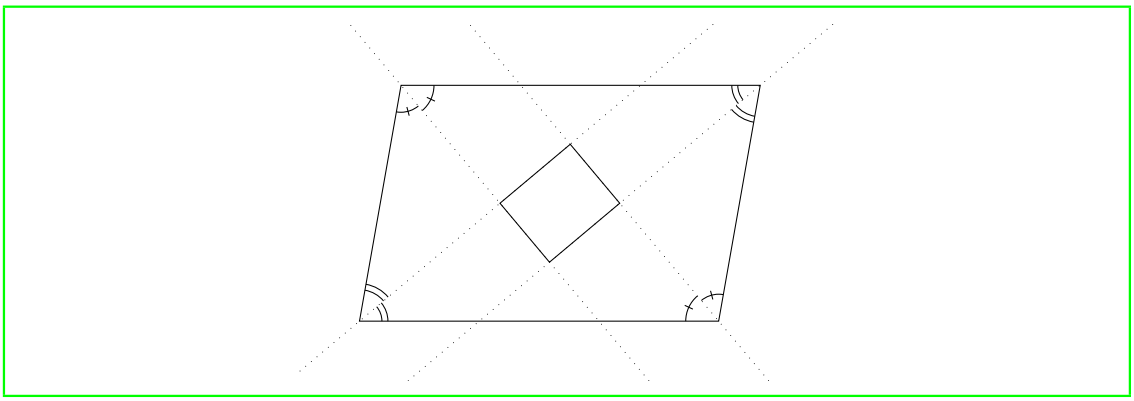


Figura 5.5. Eukleides: illustrazione per un esercizio sulla posizione del centro di massa dell'oggetto raffigurato al variare di r (il file `gravity_center_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

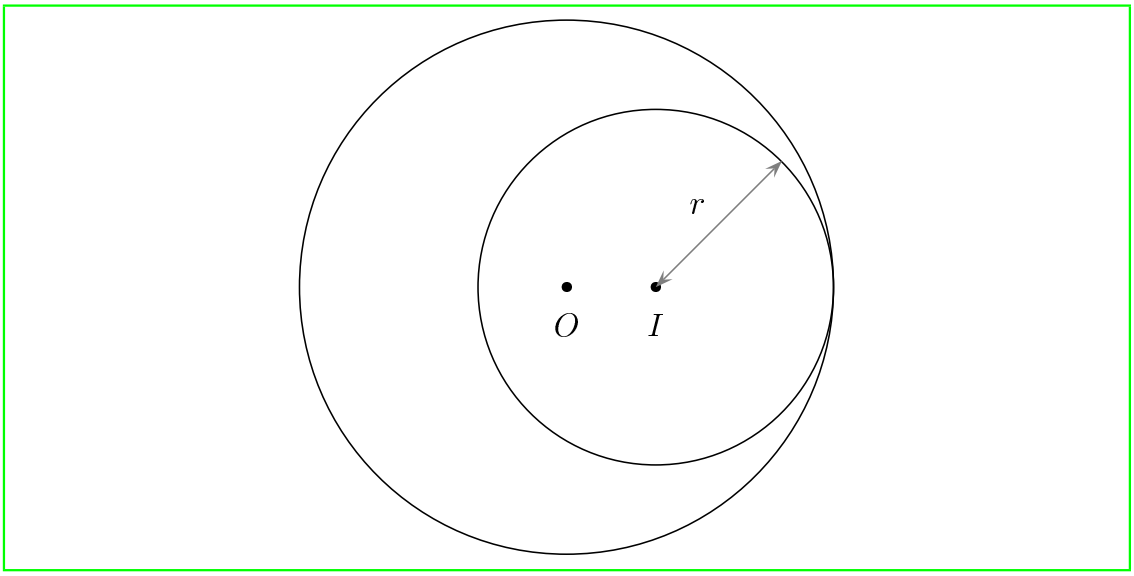


Figura 5.6. Eukleides: illustrazione relativa a un esercizio sulle omotetie (il file `intersection_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

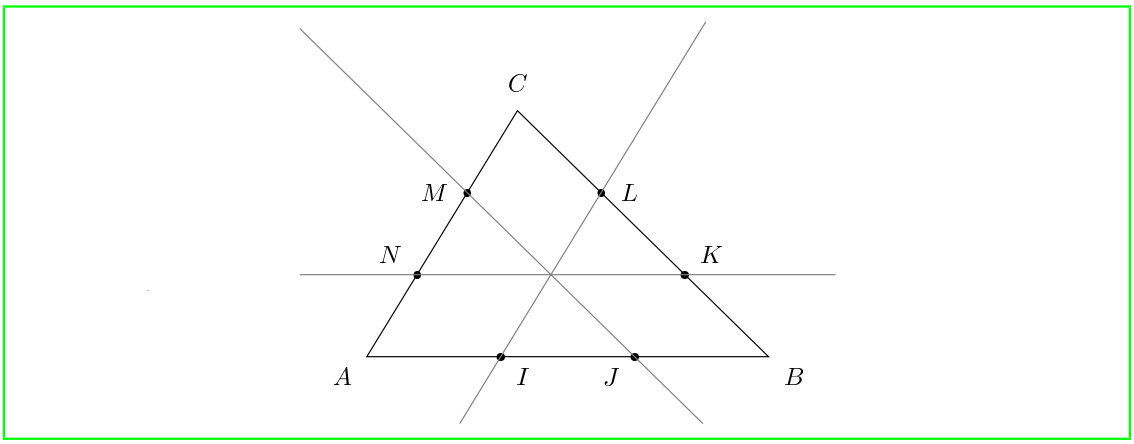


Figura 5.7. Eukleides: illustrazione relativa a un esercizio sulle omotetie (il file `lines_and_circles_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

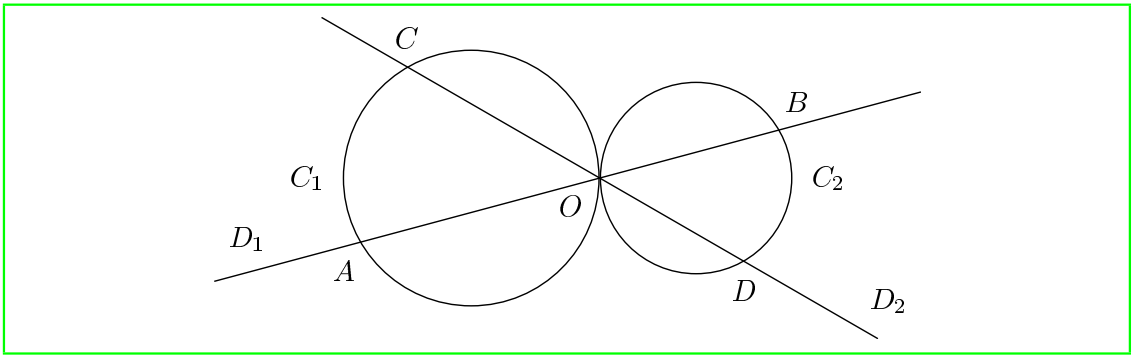


Figura 5.8. Eukleides: un classico problema sui luoghi geometrici (il file `locus_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

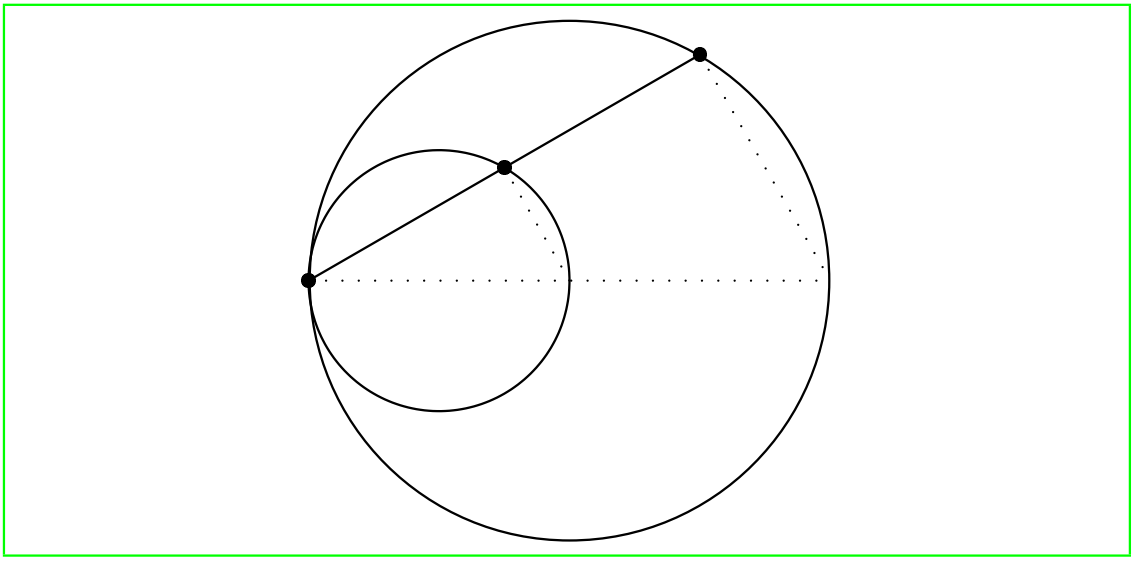


Figura 5.9. Eukleides: illustrazione del teorema di Morley (il file `morley_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

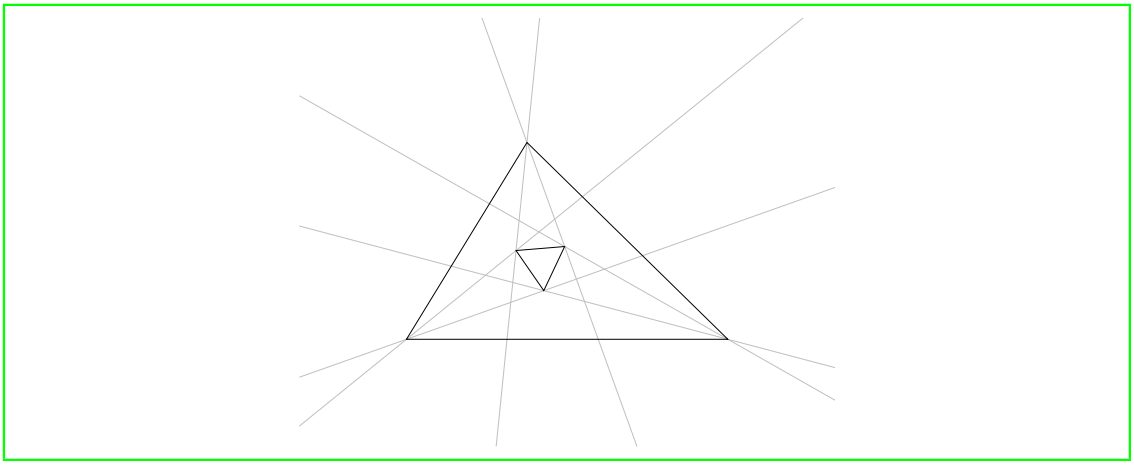


Figura 5.10. Eukleides: una proprietà dell'ortocentro (il file `orthocenter_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

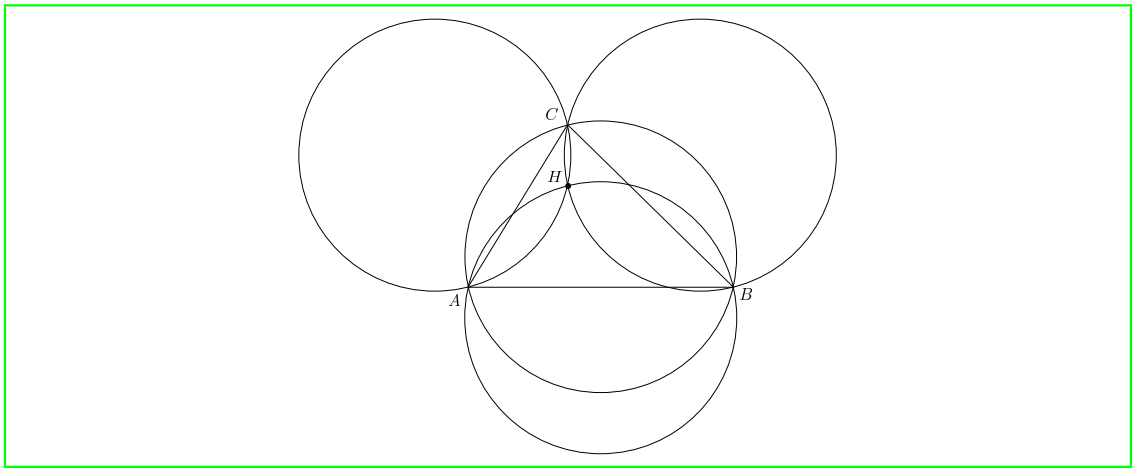


Figura 5.11. Eukleides: illustrazione relativa a un esercizio sulle omotetie (il file `parallelogram_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

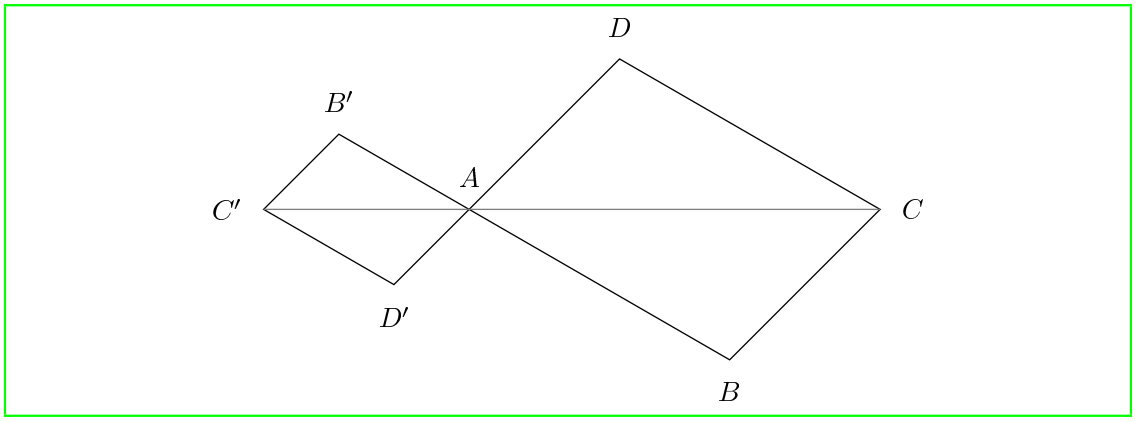


Figura 5.12. Eukleides: illustrazione del teorema di Pascal (il file `pascal_euk' contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

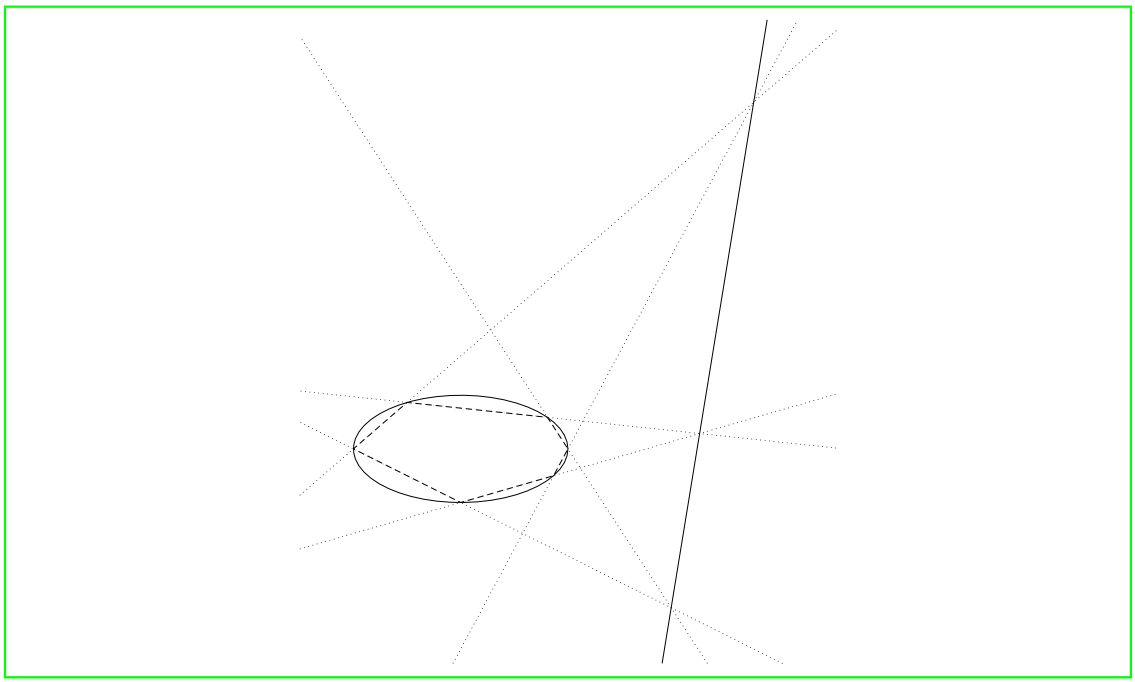


Figura 5.13. Eukleides: illustrazione per un esercizio sugli assi dei segmenti (il file `quadrilateral_euk' contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

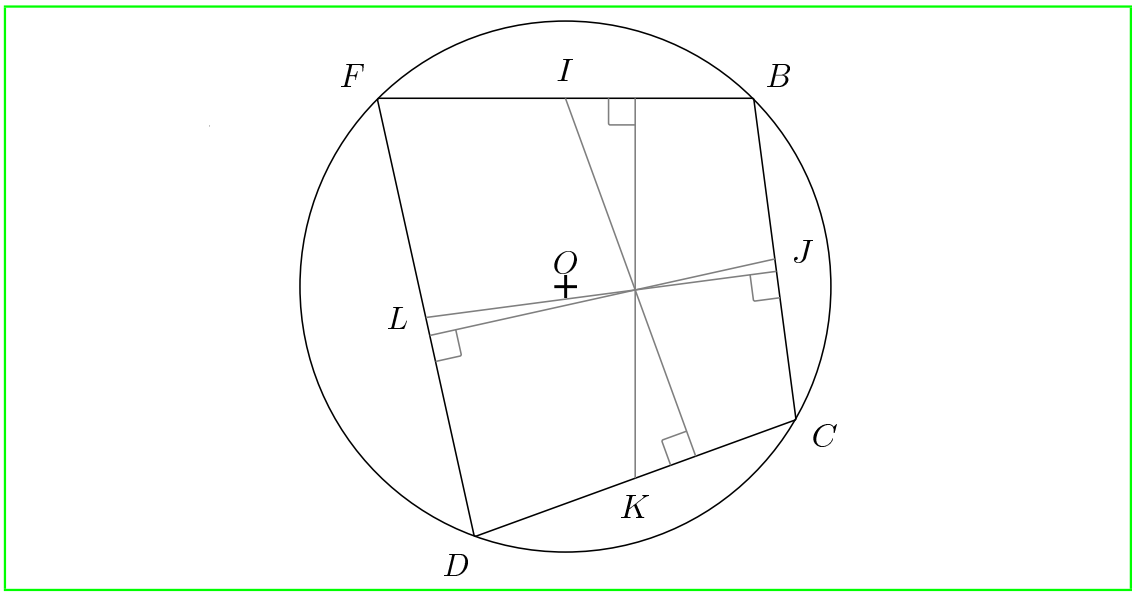


Figura 5.14. Eukleides: triangoli simili (il file `similar_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).

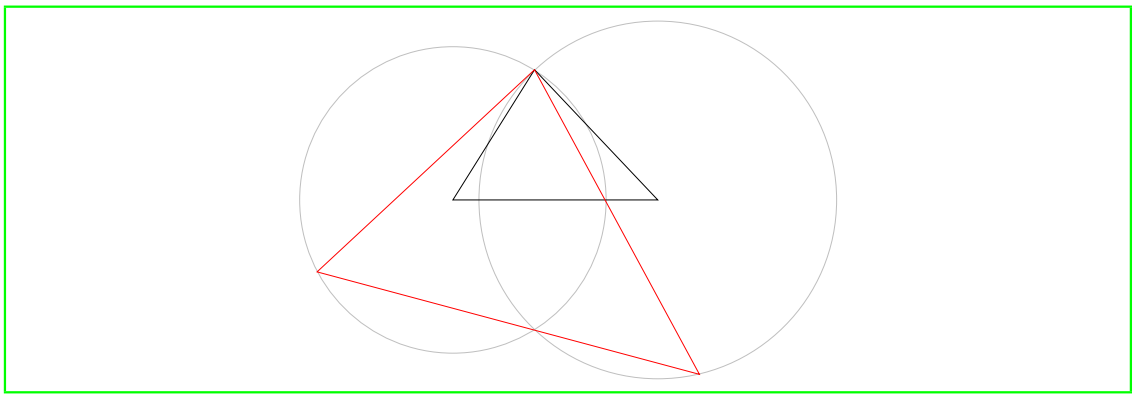
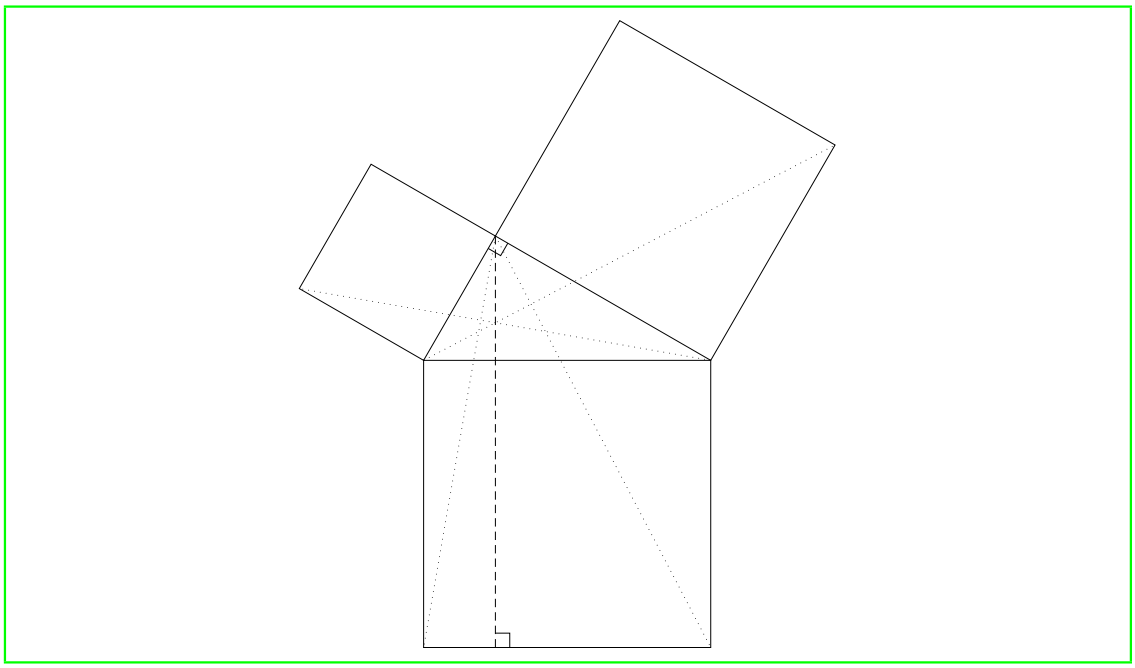


Figura 5.15. Eukleides: la classica figura di Vecten utilizzata da Euclide per dimostrare il teorema di Pitagora (il file `vecten_euk` contenente il codice Eukleides dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro).



Riferimenti e approfondimenti

- Christian Obrecht, ΕΥΚΛΕΙΔΗΣ
  ⟨*http://www.eukleides.org/*⟩
- Christian Obrecht, *Eukleides: A geometry drawing language*
  ⟨*https://www.tug.org/TUGboat/Articles/tb22-4/tb72obre.pdf*⟩
- TUG, *Welcome to the PSTricks web site*
  ⟨*http://www.tug.org/PSTricks/*⟩
- X Consortium Inc., *X Logical Font Description Conventions*
  ⟨*http://www.xfree86.org/current/xlfd.pdf*⟩
- Bill Casselman
  - *Pictures and Proofs*
    ⟨*http://www.ams.org/notices/200010/fea-casselman.pdf*⟩
  - *Mathematical Illustrations*
    ⟨*http://www.math.ubc.ca/~cass/graphics/manual/*⟩
- Alexander Bogomolny, *Pythagorean Theorem and its many proofs*
  ⟨*http://www.cut-the-knot.org/pythagoras/index.shtml*⟩

Appendici

Appendice A

Informazioni aggiuntive sul software e altre opere citate

Eukleides, IV

GNU GPL

Massimo Piai *(... circa 1975)*
⟨*pxam67 (ad) virgilio·it* ⟩