# Graphics

**Eukleides: A geometry drawing language**

Christian Obrecht

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 *PSTricks* package to draw them. I didn't want to use WYSIWYG software instead, because I wanted to keep following LaTeX's philosophy, that is: What You Mean Is What You Get. Unfortunately, *PSTricks* isn't designed for geometry at all and is rather inappropriate in many situations.

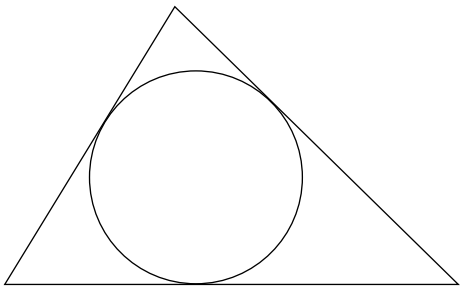
One night, I wanted to draw a triangle with an inscribed circle, so I had to compute by hand the coordinates of the center and the radius of this circle, which is quite boring. During these calculations, I realized that they could easily be done by a computer, and that gave me the idea to create Eukleides, a geometry drawing language. My goal was to make it as close as possible to what mathematics teachers would say to describe geometric figures. For instance, the former problem, written as an exercise, could be:

> Let $ABC$ be a triangle and $\mathcal{I}$ its inscribed circle. Draw $ABC$ and $\mathcal{I}$.

In Eukleides, it gives:

```
A B C triangle
I = incircle(A,B,C)
draw(A,B,C) ; draw(I)
```

Which leads to the following graphical result:



Once the design of the language was done, I wrote `eukleides`,[1] a compiler which translates Eukleides code into *PSTricks* macros. This program can run as a filter. That is, it can take a LaTeX source containing Eukleides code, and replace this code with *PSTricks* macros, producing a ready-to-TeX file.

There's also a graphical interface to the language, with additional interactive features, named

Editor's note: This article is a reiteration of the article by the same title in *TUGboat* **22**:4, pp. 334-337. Unfortunately, owing to an editorial glitch, the figures in that version were not properly displayed. We regret the mixup.

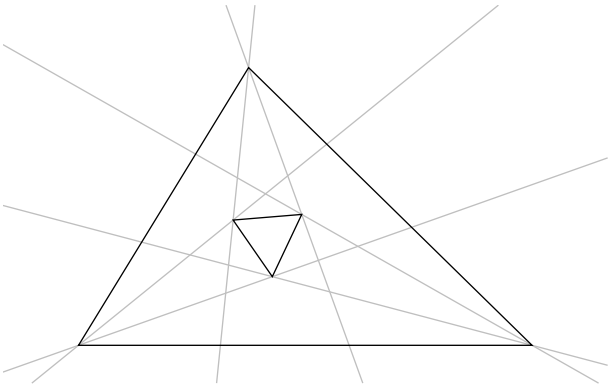
[1] It was formerly named `euklides`, but this name was already given to other geometry software.

`xeukleides`. It was first meant for classroom presentations, but it can also be seen as a tool to compose and tune some Eukleides code for later inclusion in a LaTeX source.

Both programs are released under the GNU Public License. They were developed on a GNU/Linux system, and were ported to several operating systems: NetBSD, FreeBSD, Mac OS X, MS Windows. Their source code is available from CTAN[2] or the Eukleides home page[3] (which also offers GNU/Linux and Win32 executables).

### Around Morley's triangle

As a first introduction to the Eukleides language, we'll study the source code which gives the following figure. It illustrates Morley's theorem: *The points of intersection of the adjacent trisectors of the angles of any triangle are the vertices of an equilateral triangle.*



Here is the corresponding code[4].

```
1   A B C triangle
2   a = angle(B,A,C)
3   b = angle(C,B,A)
4   c = angle(A,C,B)
5   ab = angle(vector(A,B))
6   bc = angle(vector(B,C))
7   ca = angle(vector(C,A))
8   l1 = line(A,(ab + a/3):)
9   l2 = line(A,(ab + 2*a/3):)
10  l3 = line(B,(bc + b/3):)
11  l4 = line(B,(bc + 2*b/3):)
12  l5 = line(C,(ca + c/3):)
13  l6 = line(C,(ca + 2*c/3):)
14  D = intersection(l1,l4)
15  E = intersection(l3,l6)
16  F = intersection(l2,l5)
17  color(lightgray)
18  draw(l1) ; draw(l2)
19  draw(l3) ; draw(l4)
20  draw(l5) ; draw(l6)
21  color(black)
22  draw(A,B,C) ; draw(D,E,F)
```

In Eukleides source code, a line can contain several commands (in that case, they have to be separated by semicolons). Commands are of two kinds: variable assignments and graphical commands. Among variable assignments are single assignments (see lines 2–16) and multiple assignments (line 1). A variable can store a wide variety of objects used in elementary geometry: numbers, vectors, points, lines, segments, circles, conics.

Multiple assignments are used for definitions of polygons and for some intersection determinations. The statement in line 1 defines an optimal scalene triangle such that segment $AB$ is horizontal and 6 cm long. All these characteristics can be modified by adding some optional parameters to the keyword '`triangle`'. For instance '`A B C triangle(4,5,6)`' would define a triangle $ABC$ such that $AB = 4$ cm, $BC = 5$ cm and $AC = 6$ cm.

If the desired triangle has to be of a specific kind, the simplest way is to replace '`triangle`' with '`right`', '`isosceles`' or '`equilateral`'. For instance '`A B C right(5,30:,10:)`' would define a triangle $ABC$ with an angle of 30° in $A$, a right angle in $B$ and such that segment $AB$ measures 5 cm and makes an angle of 10° with the horizontal direction. The colon character is used to distinguish angular parameters from others (like lengths).

On lines 2–7, one can see two possible usages of the function '`angle`'. In the first case (lines 2–4), it simply gives the measures of the angles in triangle $ABC$. In the second case (lines 5–7), it gives the argument of some vectors. As with many functions in Eukleides, '`angle`' can handle several kinds of arguments.

On lines 8–13 are the definitions of the trisectors of the triangle $ABC$. Since trisectors (unlike bisectors) aren't very common objects, there's no built-in function to define them. The function '`line`' is used instead. Here, the second argument is the angle that the line makes with the horizontal direction. This is not the only way to define a line: the second argument could have been a point or a vector.

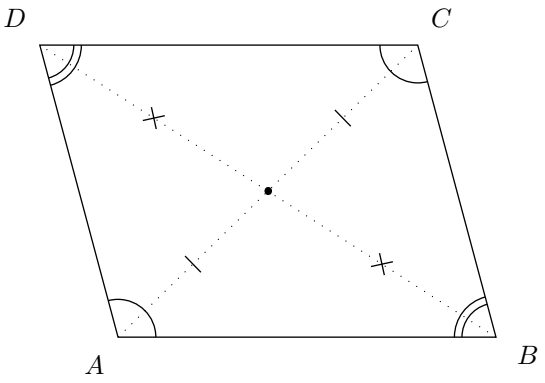
Graphical commands are of two kinds: setting commands (see lines 17 and 21) and drawing commands (see lines 18–20 and 22). To draw an object,

[2] In `/tex-archive/support/eukleides/`.
[3] At `http://perso.wanadoo.fr/obrecht/`.
[4] The numbers at the beginning of each line are not part of it.

one simply has to use the function '`draw`'. This function can take additional arguments in order to modify the aspect of the drawn object (such as '`dotted`' or '`dashed`' for lines). On line 22, the arguments of '`draw`' are a list of points: it's the way to draw polygons. Since polygons are not considered as specific objects in Eukleides, there's no need to declare $DEF$ as a triangle before drawing it.

### More graphical commands

Usually, a geometric figure doesn't contain only straight and curved lines, but also letters and some conventional marks (used to make some properties obvious). Below is a classical example of such a figure representing a parallelogram.




Here is the corresponding code.

```
1   A B C D parallelogram(5,4,105:)
2   O = barycenter(A,B,C,D)
3   frame(-2,-1,6,4.5)
4   draw(A,B,C,D) ; draw(O)
5   draw("$A$",A,-130:)
6   draw("$B$",B,-30:)
7   draw("$C$",C,50:)
8   draw("$D$",D,130:)
9   draw(segment(A,C),dotted)
10  draw(segment(B,D),dotted)
11  mark(segment(A,O))
12  mark(segment(O,C))
13  mark(segment(B,O),cross)
14  mark(segment(O,D),cross)
15  mark(B,A,D)
16  mark(D,C,B)
17  mark(C,B,A,double)
18  mark(A,D,C,double)
```

Since this figure is rather simple (from a geometrical point of view) only two assignments are needed. On line 1 is a multiple assignment which defines a parallelogram $ABCD$ such that $AB = 5$ cm, $AD = 4$ cm and $\widehat{BAD} = 105°$. On line 2, a single assignment defines $O$ as the center of parallelogram $ABCD$.

Even though Eukleides is designed in order to use as few coordinates as possible, the internal representation of the geometrical objects is based on them. By default, figures are drawn in a frame such that the lower left corner has coordinates $(-2\,;\,-2)$ and the upper right corner $(8\,;\,6)$. The function '`frame`' enables one to change these settings.

As one can see on line 4, the function '`draw`' is useful to represent single points (the default shape is a dot, but it can also be a square or a cross). This function can also be used to give names to points,[5] as in lines 5–8. Here, the first argument is a string, the second a point and the third an angular argument specifying the position of the label. This string can contain TEX code[6] such as mathematical formulas.

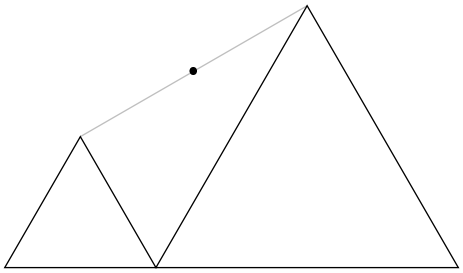
On lines 11–18 are the marking commands. It is possible to mark, in various ways, either segments (lines 11–14) or angles (lines 15–18).

### A classical locus problem

In some situations, a computer screen can be very useful to teach geometry. For instance, a locus problem becomes much easier if one can see several states of the figure. The program `xeukleides` has been developed for this. At startup, it appears as a text editor. If you type the lines below:

```
1   x interactive(2,.1,0,6,"A",right)
2   A M I equilateral(x)
3   M B J equilateral(6-x)
4   color(lightgray)
5   draw(segment(I,J))
6   color(black)
7   draw(A,M,I) ; draw(M,B,J)
8   draw(barycenter(I,J))
```

and press the escape key, the text area will be replaced by a graphical area containing the following figure:



[5] Or to put any kind of text in a specific place.

[6] This code will only be interpreted if you run `eukleides` and `latex`. With `xeukleides` it is displayed verbatim.

If you now press the right arrow key, you'll see the left triangle becoming bigger and the right one smaller. Pressing the left arrow key performs the opposite transformation. Pressing the escape key again switches back to the text editor.

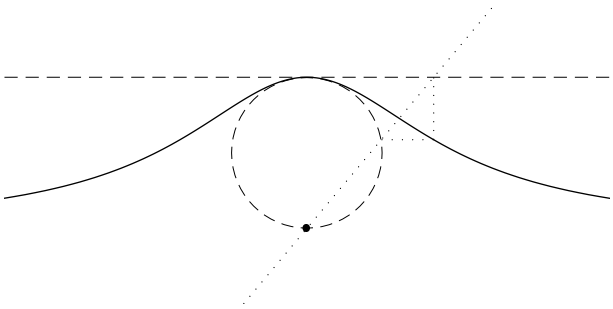
On line 1 of the source code is an interactive assignment: it allows to modify the value of the numerical variable $x$ (and consequently the figure) by pressing the arrow keys. The first argument is the initial value of $x$, the second the increment which is added to (subtracted from) $x$ every time the right (left) arrow key is pressed. The third and fourth are the optional lower and upper bound. The fifth argument has to be a string containing a single letter. It indicates the key that has to be pressed before modifying the variable.[7] This is useful when more than two variables have to be bound to the keyboard. The sixth argument is either '`right`' or '`up`'. It indicates which pair of arrow keys (right/left or up/down) is bound to the variable.

In an interactive assignment, the initial value can be modified while viewing. If you press the F1 key, the program replaces the original initial value in the source code by the last value of the variable and switches back to the text editor.

The first multiple assignment on line 2 defines an equilateral triangle $AMI$ such that segment $AM$ is horizontal and $x$ cm long. The second assignment defines an equilateral triangle $MBJ$ such that segment $MB$ is horizontal and $6-x$ cm long. A specific feature of polygonal assignments is used here: if the first variable is already in use (and contains a point) its content remains the same (if not, the variable is set to the origin). This implies that segment $AB$ has a constant length of 6 cm and that $M$ belongs to $AB$.

### Drawing curves

In elementary geometry, the most usual curves are conics. Eukleides provides a large number of function to define and handle these objects. For less common curves, there's the '`trace`' command. For instance, the figure below illustrates the geometrical definition of a cubic curve known as the *Witch of Agnesi*.



This curve is obtained by drawing a line from the origin through the dashed circle, then picking the point with the $x$ coordinate of the intersection whith the dashed line and the $y$ coordinate of the intersection with the circle. Here is the corresponding code:

```
1   frame(-4,-1,4,3)
2   O = point(0,0)
3   c = circle(point(0,1),1)
4   l = line(point(0,2),0:)
5   trace(t,.1,179.9){
6   L = line(O,t:)
7   O M intersection(L,c)
8   P = intersection(L,l)
9   point(abscissa(P),ordinate(M))}
10  t = 50
11  L = line(O,t:)
12  O M intersection(L,c)
13  P = intersection(L,l)
14  N = point(abscissa(P),ordinate(M))
15  draw(O)
16  style(dotted)
17  draw(segment(M,N))
18  draw(segment(P,N))
19  draw(L)
20  thickness(.5) ; style(dashed)
21  draw(c) ; draw(l)
```

On lines 2–4 we create the objects which are needed to define the curve. Lines 5–9 are related to the '`trace`' command. They are based on the geometrical definition above. Line 5 tells variable $t$ to scan the numbers between[8] 0.1 and 179.9. A line-circle intersection can lead to two points, hence in Eukleides a multiple assignment (like the one at line 7) is used to obtain these points. Since lines are implicitly directed, in the present case the first assigned point will always be $O$ and the second, the wanted point. The last line (line 9) contains a point

[7] Since the program starts viewing in state "A", there's no need here to press this key.

[8] These bounds are chosen in order to avoid 0 and 180, which are invalid and may cause spurious lines to appear.

valued expression. This is the point which will be drawn for each value of $t$.

To draw this curve, it would also be possible to use parametric representation. Nevertheless, in the present case the geometrical definition is more appropriate because the same piece of code can be used again (in lines 11–14) to produce an example of the construction.

The last part (lines 15–21) contains the drawing commands. The setting command '`style`' changes the default aspect of drawn objects. This may sometimes shorten the code.

### Conclusion

In my humble opinion, Eukleides is now mature enough to be considered by TEX users as an effective way to create geometric figures. As a matter of fact, the language is sufficiently powerful to describe almost any figure which can be seen in an elementary geometry textbook.

My aim is now to enhance Eukleides with features such as tests, loops, and user-defined functions. Since I did not anticipate this when I started the project, I'll have to rewrite large parts of the programs. This is a long-term undertaking, so I'll soon stop working on the present versions of `eukleides` and `xeukleides`.

◇ Christian Obrecht
Le Monsard
71960 Bussieres
France
christian.obrecht@wanadoo.fr
http://perso.wanadoo.fr/obrecht/