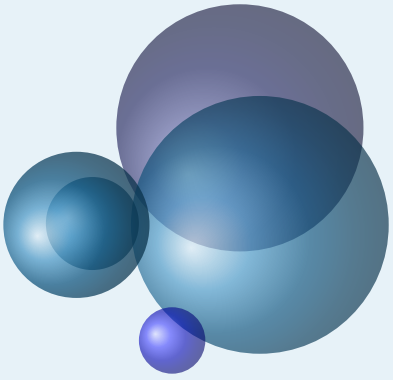
tkz-elements 1.20b

AlterMundus

tkz-elements 1.20b

Euclidean Geometry



Alain Matthes

November 10, 2023 Documentation V.1.20b

http://altermundus.fr

tkz-elements

Alain Matthes

AlterMundus

☞ This document brings together some notes about **tkz-elements**, the first version of a library written in lua, allowing to make all the necessary calculations to define the objects of a Euclidean geometry figure. You need to compile with LuaLaTeX.
With tkz-elements, the definitions and calculations are only done with lua.
The main possibility of programmation proposed is oriented "object programming" with object classes like point, line, triangle, circle and ellipse. For the moment, once the calculations are done, it is tkz-euclide or TikZ which allows the drawings.
I discovered Lua and object-oriented programming when I created this package, so it's highly probable that I've made a few mistakes. If you'd like to participate in the development of this package or give me advice on how to proceed, please contact me via my email.
This version 1.20b is a beta version.
English is not my native language so there might be some errors.

☞ Acknowledgements : I received much valuable advice, remarks, corrections from **Nicolas Kisselhoff**, **David Carlisle**, **Roberto Giacomelli** and **Qrrbrbirlbel**.

☞ I would also like to thank **Eric Weisstein**, creator of MathWorld.

☞ You can find some examples on my site: altermundus.fr. under construction!

Contents

1 Presentation

I created tkz-euclide to give math teachers and students a tool to draw Euclidean geometry figures. This package was created as an interface based on TikZ, itself based on TEX. The only problem encountered with this method is, in complicated cases, the lack of precision of the calculations made by TEX.
To remedy this lack of precision, I first introduced the package fp, then the package xfp. Lately, with the arrival of luaLATEX, I have been able to add a **Lua** option whose goal was to perform some calculations with **Lua**. It is much easier to program mathematics with **Lua** than with TEX so having prepared tkz-euclide, by separating the first parts of definitions and calculations from the drawing part, the idea of carrying out the first parts with only **Lua** was necessary.
I had received some examples of programming with **Lua** from Nicolas Kisselhoff and I took many of his ideas.
The essential principles of figure construction with **tkz-euclide** are kept: definition, calculations, tracings, labels as well as the step-by-step programmation, corresponding to a construction with a ruler and a compass.
Then, I read an article by Roberto Giacomelli on object programming based on the **Lua** and Ti*k*Z tools. This was my second source of inspiration. Not only could the programming be done step-by-step, but the introduction of objects allowed the link between the code and the geometry. The code becomes more readable, the code is more explicit and better structured but it is less concise. Finally, as the problem of precision disappeared, I came to appreciate more and more what I was getting with object programming. So I've tried to develop this programming as much as possible.
In order to organize and maintain all the functions and methods, I chose to use classes of objects whose main ones are **point**, **line**, **triangle**, **circle** and **ellipse**.

1.1 With Lua

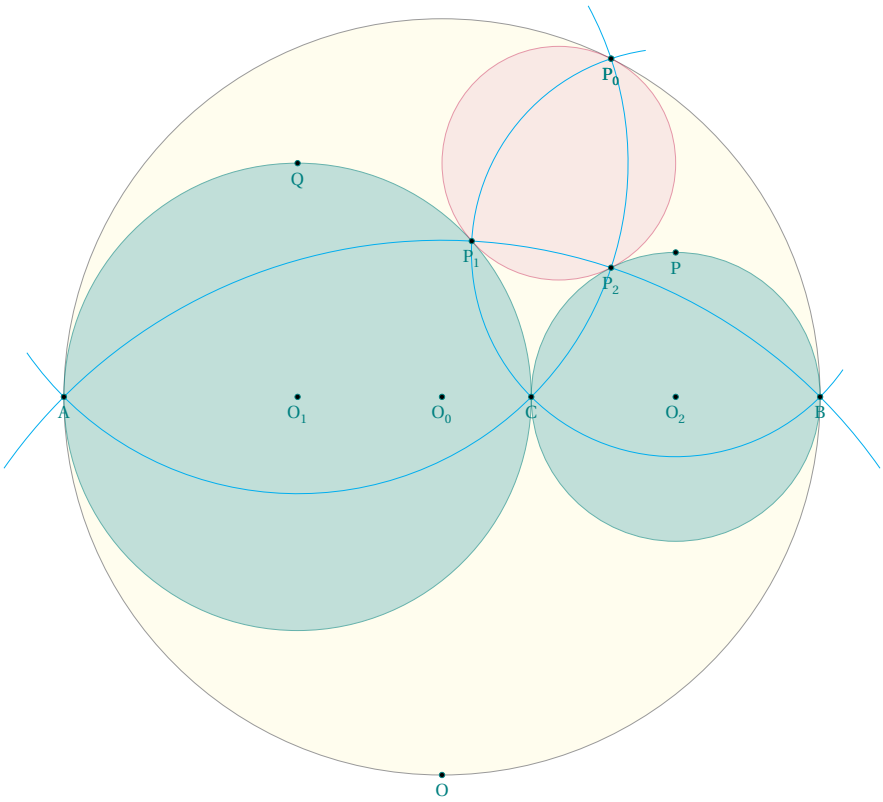
The purpose of tkz-elements is simply to calculate dimensions and define points. This is done in Lua. You can think of tkz-elements as a kernel that will be used either by tkz-euclide or by TikZ, see MetaPost. Definitions and calculations are done inside the environment **tkzelements**, this environment is based on **luacode**. To improve precision, if it's necessary to modify the scale of your figure, it's best to change the "scale" variable at the start of the code placed in environment **tkzelements**. Then we need to be able to transfer the coordinates of the points to a package that can use them, in this case tkz-euclide or TikZ.

1.2 The main process : from tkz-elements (lua) to tkz-euclide (TikZ) with LuaLATEX.

When all the points necessary for the drawing are obtained, they must be transformed into **nodes** so that TikZ or tkz-euclide can draw the figure. This is done through the macro **\tkzGetNodes**. This macro browse all the elements of the table z using the key (in fact the name of the point) and retrieves the values associated with it, i.e. the coordinates of the point (node).
A point A is defined and stored in tkz-elements as follows: z.A is an element of a table z which contains the coordinates of A in the form of a complex number (very useful for calculations). Then a macro, **\tkzGetNodes**, transforms all elements of table z into nodes usable by tkz-euclide or TikZ. For another package, you'll need to adapt **\tkzGetNodes**.

1.3 Complete example: Pappus circle

The figure:



1.3.1 The code

```
% !TEX TS-program = lualatex
\documentclass{article}
\usepackage{tkz-euclide}
\usepackage{tkz-elements}
\begin{document}

\begin{tkzelements}
   z.A      = point: new (0 , 0)
   z.B      = point: new (10 , 0)
   L.AB     = line:  new ( z.A, z.B)
   z.C      = L.AB:  gold_ratio ()
   L.AC     = line:  new ( z.A, z.C)
   L.CB     = line:  new ( z.C, z.B)
   L.AB     = line:  new ( z.A, z.B)
   z.O_0    = L.AB.mid
   z.O_1    = L.AC.mid
   z.O_2    = L.CB.mid
   C.AB     = circle: new ( z.O_0, z.B)
   C.AC     = circle: new ( z.O_1, z.C)
   C.CB     = circle: new ( z.O_2, z.B)
   z.P      = C.CB.north
   z.Q      = C.AC.north
   z.O      = C.AB.south
   z.c      = z.C : north (2)
```

```
25    C.PC     = circle: new ( z.P, z.C)
26    C.QA     = circle: new ( z.Q, z.A)
27    z.P_0    = intersection (C.PC,C.AB)
28    z.P_1    = intersection (C.PC,C.AC)
29    _,z.P_2  = intersection (C.QA,C.CB)
30    T        = triangle: new ( z.P_0, z.P_1, z.P_2)
31    z.O_3    = T.circumcenter
32 \end{tkzelements}
33
34 \begin{tikzpicture}
35   \tkzGetNodes
36   \tkzDrawCircle[black,fill=yellow!20,opacity=.4](O_0,B)
37   \tkzDrawCircles[teal,fill=teal!40,opacity=.6](O_1,C O_2,B)
38   \tkzDrawCircle[purple,fill=purple!20,opacity=.4](O_3,P_0)
39   \tkzDrawArc[cyan,delta=10](Q,A)(P_0)
40   \tkzDrawArc[cyan,delta=10](P,P_0)(B)
41   \tkzDrawArc[cyan,delta=10](O,B)(A)
42   \tkzDrawPoints(A,B,C,O_0,O_1,O_2,P,Q,P_0,P_0,P_1,P_2,O)
43   \tkzLabelPoints(A,B,C,O_0,O_1,O_2,P,Q,P_0,P_0,P_1,P_2,O)
44 \end{tikzpicture}
45 \end{document}
```

1.3.2 Some explanations

Half of the code concerns the direct creation of objects (`point: new` , `circle: new` etc.)
Then points are obtained using object attributes like `L.AB.mid` which determines the middle of the segment [AB] or `C.AB.south` which allows to obtain the south pole of the circle with center A passing through B.
About the naming of objects, see the section 2

- Lines 8 and 9 create two fixed points A and B.
- Line 10 these two points are used to create the line.
- Line 11 the gold method creates a point C which divides the segment [AB] according to the golden ratio.
- Lines 12, 13 and 14 creation of 3 new lines.
- Lines 15, 16 and 17 creation of 3 points: the midpoints of the previous segments. These points are predefined and it is enough to use the attributes of the lines. Another possibility is to use the triangle T.ABC with the method : `medial_tr`
- Lines 18, 19 et 20 creation of three circles.
- Line 21 creation of the south pole P of the circle of center C passing through B. It is also a predefined point and we still use an attribute.
- Lines 22 and 23 we create two new poles: north and south for two other circles.
- Lines 24 and 25 creation of two circles.
- Lines 26 and 27 search for intersections of two circles.
- Lines 28 and 29 we use a triangle to search for its circum center.

2 Writing Convention

Writing conventions for real numbers are the same as for Lua. The package introduces complex numbers (see 4.2) in order to associate them with points. `z1 = point (1,2)` designates the affix `z1=1+2i`, `z1` is a complex number, but here it's not a object point (related to objects in object-oriented programming).
`z.A = point : new (1,2)` creates an object point A with coordinates 1 and 2, which are stored as an affix in the table `z`. The key is A and the value `1+2i` (or (1,2) if you prefer.)

2.1 Assigning a Name to a Point

The basic object is the point which belongs to the class point. A point can be created in two ways: either directly `z.A = point : new (1,2)`, or as the result of a function or a method.
Currently the only obligation is to store the points in the table `z` [1] if you want to use them in Ti*k*Z or `tkz-euclide`.
If it is a point which will not be used, then you can designate it as you wish by respecting the conventions of Lua.
More generally, with `z.A = point: new (a,b)`, we define a complex number `z.A = a+bi`, where a and b are the coordinates of point A. The name A of a point is the key associated with a complex number noted `z.A` whose affix is `a+ib`, with a and b unique coordinates of A.
There are two cases. Either you want to keep the point to use it with the package that will allow you to draw, or the point is intermediate and you can abandon it.

In the second case, simply don't store the point in the `z` table. For example, `A = point : new (1,2)` defines a point A that will not be transferred. Another way of avoiding a transfer is to assign `nil` to the point: `z.A = nil` before exiting the environment **`tkzelements`**.
In the first case, you must store the point in the table `z`. The points which occur in the environment **`tkzelements`** must respect a convention which is `z.name` such that `name` will be the name of the corresponding **`node`**. [2]
What are the conventions for designating `name`? You have to respect the Lua conventions in particular cases.

1. The use of prime is problematic. If the point name contains more than one symbol and ends with `p` then when passing into TikZ or `tkz-euclide`, the letters `p` will be replaced by `'` using the macro **`\tkzGetNodes`**;

2. One possibility, however, in order to have a more explicit code is to suppose that you want to designate a point by "euler". It would be possible for example to write `euler = ...`, and at the end of the code for the transfer, `z.E = euler`. It is also possible to use a temporary name `euler` and to replace it in Ti*k*Z. Either at the time of placing the labels, or for example by using `pgfnodealias{E}{euler}`. This possibility also applies in other cases: prime, double prime, etc.

☞ The syntax `z.name` is important, it allows to store the data of the point in a table named `z`, which will be used for the transition to the program that will perform the plots and place the labels.
Point names: You can use the lower case and unaccented upper case.
The names of the points follow the rules of convention for writing identifiers in lua. The name starts with a letter "A to Z" or "a to z" or an underscore "_" followed by zero or more letters, underscores, and digits (0 to 9).
You should avoid identifiers starting with an underscore followed by one or more uppercase letters.
In general, the points used in geometry are lower or upper case letters. The use of underscore is also frequent. So it is possible to write `z.T_a`.
☞ Another use in geometry that can be used with Ti*k*Z is the apostrophe. Unfortunately, the apostrophe is necessary to use strings. There are several ways to get around this difficulty. The simplest one, which keeps the syntax consistent, is that if you want to use `A'` as a point name, you can use `Ap`. When switching to Ti*k*Z `Ap` will be replaced by `A'`. All names of more than one character ending in `p` will have the letter `p` replaced by an apostrophe `'`.

1 To place the point M in the table, simply write `z.M` = ...or `z["M"]` = ...
2 However, that a point can be deleted with `z.name = nil`.

2.2 Assigning a Name to Other Objects

You have the choice to give a name to objects other than points. That said, it is preferable to respect certain rules in order to make the code easier to read. I have chosen for my examples the following conventions: first of all I store the objects in tables: `L.name` for lines and segments, `C.name` for circles, `T.name` for triangles, `E.name` for ellipses.
For lines, I use the names of the two points. So if a line passes through points A and B, I name the line `L.AB`. It is possible if only one line is named to simply name it `L` but this should be avoided so as not to create bad practices.
For circles, I name `C.AB` the circle of center A passing through B, but siomething like `C.euler` or `C.external` is fine.
For triangles, I name `T.ABC` the triangle whose vertices are A, B and C.
For ellipses, I name `E.ABC` the ellipse with center A through vertex B and covertex C.

2.3 Writing conventions for attributes, methods.

You must use the conventions of Lua, so

- To obtain an attribute, for all objects, the convention is identical: `object.attribute`. For example, for the point A we access its abscissa with `z.A.re` and its ordinate with `z.A.im`; as for its type we obtain it with `z.A.type`. To get the south pole of the circle `C.OA` you need to write: `C.OA.south`.
- To use a method such as obtaining the incircle of a triangle ABC, just write

  `z.i,z.i_a,z.i_b,z.i_c = T.ABC : in_circle ()` (you get the center and its projections on the sides), but to get only the incenter you write `T.ABC.incenter`.
- Some methods need a parameter. For example, to know the distance between a point C to the line (A, B) we will write

  `d = L.AB : distance (z.C).`
- Use the underscore to store a result you don't want to use. If you only need the second point of an intersection between a line and a circle, you would write

  `_,z.J = intersection (L.AB , C.OC).`

3 How to use

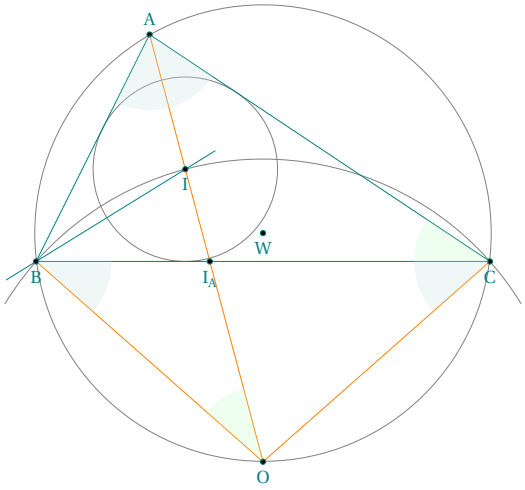
You need to load either Ti*k*Z or tkz-euclide if you want to use these packages to get your plots, then you need to load tkz-elements.
Loading tkz-elements allows you to use the elements environment. However, you can use the macro \directlua or the luacode environment (the package luacode is loaded automatically).

3.1 Environment elements

You must first create an environment **tkzelements**, then in this environment you create the fixed points of your figure. (For the moment, you place the environment elements before the environment tikzpicture but it is possible to place it inside the latter.)
In the following example, three fixed points are used (lines 8, 9 and 10), the first object **point** A is created by
`z.A = point: new (2 , 4).`
Then we create a **triangle** (line 11)
`T.ABC = triangle: new (z.A,z.B,z.C)`
Once the triangle has been created, you can either use its attributes or a method associated with the triangle object. In all cases, the result is either a number or a new object.
In order to obtain the circle inscribed in this triangle, we use a method of the **triangle**. We obtain a new object, a circle, which I call `C.Ii`. A circle is defined by its center and a point through which it passes. I is the center and i is the projection of this center on the side opposed to the first point of the triangle.
There are several ways of obtaining these two points. As in the example, `get_points` is a function that extracts the points from the object given as an argument. You can also use the circle attributes: `z.I = C.Ii.center` and `z.i = CIi.through`. The first method is obviously simpler, but less explicit. Another possibility would be to obtain the center as an attribute of the triangle `z.I = T.ABC.incircle` and then project this center onto one of the sides, for example `z.i = T.ABC.bc : projection (z.I)`. An explanation is in order: `T.ABC.bc` is the straight line defined by the last two points of the triangle (see triangle attributes 8).
Then we look for the circumscribed circle, which is obtained using a method of the triangle object. We define the circumscribed **circle** named `C.WA`.
`C.WA = T.ABC : circum_circle ()` (line 16)
`z.W,_ = get_points (C.WA)` is used to get the center but possible is
`z.W = T.ABC.circumcircle`
The south pole of the circle with center W passing through A is given by an attribute of the object **circle**.
We finally look for the intersection of two lines.
Line 17 an object line is created. Line 18 I name a line object already created (it's a triangle attribute).
All points are defined and stored in the z table. The macro \tkzGetNodes retrieves the keys and associated values from this table and creates the nodes to be used by Ti*k*Z or tkz-euclide.

```
1  % !TEX TS-program = lualatex
2  \documentclass{article}
3  \usepackage{tkz-euclide}
4  \usepackage{tkz-elements}
5  \begin{document}
6
7  \begin{tkzelements}
8     z.A     = point: new (2 , 4)
9     z.B     = point: new (0 , 0)
10    z.C     = point: new (8 , 0)
11    T.ABC   = triangle: new (z.A,z.B,z.C)
12    C.Ii    = T.ABC: in_circle ()
13    z.I,z.i = get_points (C.Ii)
14    C.WA    = T.ABC : circum_circle ()
15    z.W,_   = get_points (C.WA)
```

```
   z.O     = C.WA.south
   L.AO    = line: new (z.A,z.O)
   L.BC    = T.ABC.bc
   z.I_A   = intersection (L.AO,L.BC)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(W,A I,i)
   \tkzDrawArc(O,C)(B)
   \tkzDrawPolygon(A,B,C)
   \tkzDrawSegments[new](A,O B,O C,O)
   \tkzDrawLine(B,I)
   \tkzDrawPoints(A,B,C,I,I_A,W,O)
   \tkzFillAngles[green!20,opacity=.3](A,O,B A,C,B)
   \tkzFillAngles[teal!20,opacity=.3](O,B,C B,C,O B,A,O O,A,C)
   \tkzLabelPoints(I,I_A,W,B,C)
   \tkzLabelPoints[above](A)
\end{tikzpicture}
\end{document}
```



3.2 Fom Lua to tkz-euclide or TikZ

3.2.1 Points transfer

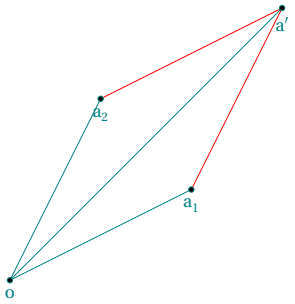
We use an environment **tkzelements** outside an environment **tikzpicture** which allows us to carry out all the necessary calculations, then we launch the macro **\tkzGetNodes** which transforms the affixes of the table **z** into a list of **Nodes**. It only remains to draw.
Currently the drawing program is either Ti*k*Z or tkz-euclide. You have the possibility to use another package to trace but for that you have to create a macro similar to **\tkzGetNodes**. Of course, this package must be able to store the points as does Ti*k*Z or tkz-euclide.

```
\def\tkzGetNodes{\directlua{%
   for K,V in pairs(z) do
      local KS,n,sd,ft
      KS = tostring(K)
      n = string.len(KS)
      if n >1 then
      _,_,ft, sd = string.find( KS , "(.+)(.)" )
```

```
     if sd == "p" then   K=ft.."'" end
       end
  tex.print("\\coordinate ("..K..") at ("..V.re..","..V.im..") ;\\\\")
end}
}
```

The environment **tkzelements** allows to use the underscore _ and the macro **\tkzGetNodes** allows to obtain names of nodes containing **prime**. (see the next example)

```
\begin{tkzelements}
   scale = 1.2
   z.o   = point: new (0,0)
   z.a_1 = point: new (2,1)
   z.a_2 = point: new (1,2)
   z.ap  = z.a_1 + z.a_2
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawSegments(o,a_1 o,a_2 o,a')
   \tkzDrawSegments[red](a_1,a' a_2,a')
   \tkzDrawPoints(a_1,a_2,a',o)
   \tkzLabelPoints(o,a_1,a_2,a')
\end{tikzpicture}
```



3.2.2 Other transfers

Sometimes it's useful to transfer angle or length measurements. For this purpose, I have created the function (see 10.0.5) `set_lua_to_tex (list)`

4 Classes and objects

4.1 Objects

The object types currently defined and used are **point**, **line**, **circle**, **triangle** and **ellipse**.
They can be created directly using the method `new` and, with the exception of the `point` class, by giving points. Objects can also be obtained by applying methods to other objects. For example, `T.ABC : circum_circle ()` creates an object **circle**. Some object attributes are also objects, such as `T.ABC.bc` which creates the object **line**, a straight line passing through the last two points defining the triangle.
Attributes are accessed using the classic method, so `T.pc` gives the third of the triangle and `C.center` gives the center of the circle, but I've added a `get_points` function that returns the points of an object. This applies to straight lines (pa and pc), triangles (pa, pb and pc) and circles (center and through).
Example: `z.O,z.T = get_points (C)` recovers the center and a point of the circle.

4.2 tkz_elements_point.lua: complex numbers library

Unless you want to create your own functions, you won't need to know and use complex numbers. However, in some cases it may be useful to implement some of their properties.
`za = point (1,2 )` and `zb = point (1,-1)` define two affixes which are `za = 1+2i` and `zb = 1-i`. The notation may come as a surprise, as I used the term "point". The aim here was not to create a complete library on complex numbers, but to be able to use their main properties in relation to points. I didn't want to have two different levels, and since a unique connection can be established between the points of the plane and the complexes, I decided not to mention the complex numbers! But they are there.
The above entries must not be confused with the following ones :
`z.a = point : new (1,2)` and `z.b = point : new (1,-1)` .
Here we define two "point" objects that are in the form of an affix in the `z` table. `z.a = za` only if `scale = 1` and `z.a` creates a point named `a`. This point is an object and you can use its attributes, as well as methods referring to points.
That said, `z.a` is indeed an affix, and it's possible to use the methods and metamethods associated with complex numbers.

Metamethods	Application
__add(z1,z2)	`z.a + z.b` –> affix
__sub(z1,z2)	`z.a - z.b` –> affix
__unm(z)	`- z.a` –> affix
__mul(z1,z2)	`z.a * z.b` –> affix
__concat(z1,z2)	`z.a .. z.b` –> dot product = real number [a]
__pow(z1,z2)	`z.a ^ z.b` –> determinant = real number
__div(z1,z2)	`z.a / z.b` –> affix
__tostring(z)	tex.print(tostring(z)) –> displays the affix
__tonumber(z)	tonumber(z) –> affix or nil
__eq(z1,z2)	eq (z.a,z.b) –> boolean

a If O is the origin of the complex plan, then we get the dot product of the vectors $\overrightarrow{Oa}$ and $\overrightarrow{Ob}$

Methods	Application
`conj(z)`	`z.a : conj()` –> affix (conjugate)
`mod(z)`	`z.a : mod()` –> real number = modulus `z.a`
`abs (z)`	`z.a : abs()` –> real number = modulus
`norm (z)`	`z.a : norm()` –> norm (real number)
`arg (z)`	`z.a : arg()` –> real number = argument of z.a (in rad)
`get(z)`	`z.a : get()` –> re and im (two real numbers)
`sqrt(z)`	`z.a : sqrt()` –> affix

The class is provided with two specific metamethods.

– Since concatenation makes little sense here, the operation associated with `..` is the scalar or dot product. If `z1 = a+ib` and `z2 = c+id` then

  `z1..z2 = (a+ib) .. (c+id) = (a+ib) (c-id) = ac+bd + i(bc-ad)`

  There's also a mathematical function, `dot_product`, which takes three arguments. See example 10.0.8

– With the same idea, the operation associated with `^` is the determinant i.e.

  `z1 ^ z2 = (a+ib) ^ (c+id) = ad - bc` From `(a-ib) (c+id) = ac+bd + i(ad - bc)` we take the imaginary part.

5 Class point

The first one is the base of the package, it is the class `point`. This class is hybrid in the sense that it is as much about points of a plane as complex numbers. The principle is the following: the plane is provided with an orthonormal basis which allows us to determine the placement of a point using its abscissa and ordinate coordinates; in the same way any complex number can simply be considered as a pair of real numbers (its real part and its imaginary part). We can then designate the plane as the complex plane, and the complex number x + iy is represented by the point of the plane with coordinates (x, y). Thus the point A will have affix z.A. This affix will be with Lua stored in the table z in the form of a couple of real numbers (x_A, y_A). The first element being the real part of the complex, and the second the imaginary part. There will be an identification between the and its corresponding affix. More precisely, for Lua the name of the point is the key that allows the table to designate the pair of reals associated to the affix.
Writing `z.A = point: new (2,3)` creates an object of class point where `z.A = 2+3i`.
The creation of a point is done using the following method, but there are other possibilities. If a scaling factor has been given, the method takes it into account.

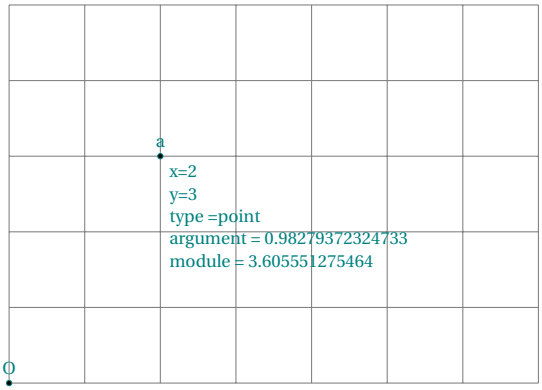
5.1 Attributs of a point

`z.A = point: new (1,2)`. The point A has coordinates x = 1 and y = 2. If you use the notation `z.A` then A will be the reference of a node in Ti*k*Z or in `tkz-euclide`.
This is the creation of a fixed point with coordinates 1 and 2 and which is named A. The notation `z.A` indicates that the coordinates will be stored in a table noted z (reference to the notation of the affixes of the complex numbers) that A is the name of the point and the key allowing access to the values.

Attributes	**Application**
re	z.a.re –> z.a = point:new (2,3) x = z.a.re -> x=2
im	z.a.im –> z.a = point:new (2,3) y= z.a.im -> y=3
type	z.a.type –> z.a.type = point
argument	z.a.argument –> z.a.argument = 0.78539816339745 if z.a = 1+i
module	z.a.module –> z.a.module = 1.4142135623731 if z.a = 1+i

5.1.1 Example:point attributes

```
\begin{tkzelements}
   z.O   = point: new (0 , 0)
   z.a   = point: new (2, 3)
   x     = z.a.re
   y     = z.a.im
   ty    = z.a.type
   arg   = z.a.argument
   m     = z.a.modulus
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzInit[ymax=4,xmax=7]
   \tkzGrid
   \tkzDrawPoints(a,O)
   \tkzLabelPoints[above](O,a)
   \tkzLabelPoint[below](a){
   \begin{minipage} {5cm}
   x=\tkzUseLua{x}\\ y=\tkzUseLua{y}\\ type =\tkzUseLua{ty}\\ argument = \tkzUseLua{arg} \\ module = \tkzUseLua{m
    \end{minipage}}
\end{tikzpicture}
```



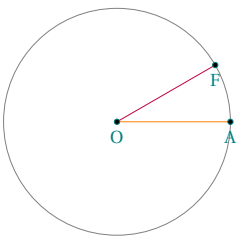
5.2 Methods of the class point

The methods described in the following table are standard. You'll find them in most of the examples at the end of this documentation. The result of the different methods presented in the following table is a **point**. A word about methods beginning with `set_`. The purpose is to be able to apply a method to a list of points.

Methods	Application
Points	
new (a,b)	`z.a = point : new(1,2)` –> affix z.a=1+2i
polar (radius, phi)	`z.a = point : polar(1,math.pi/3)` –> affix $\cos(\pi/3)+\sin(\pi/3)$i
polar_deg (radius, phi)	phi in deg –> polar coordinates phi deg
north(d)	see 12.48 d distance to the point 1 if empty
south(d)	
east(d)	
west(d)	
symmetry(z)	`z.c = z.a:symmetry(z.b)` –> c symmetry of b with respect to a
set_symmetry (...)	`z.bp,z.cp = z.a : symmetry(z.b,z.c)` –> list of points
rotation(angle , pt)	`z.c = z.a : rotation (math.pi/2,z.b)` –> affix; rotation center a
set_rotation (angle,...)	
homothety(k,pt)	`z.c = z.a : homothety (2,z.b)`
set_homothety (k,...)	
normalize()	`z.b = z.a: normalize ()` –> z.b = 1 and z.a = k× z.b

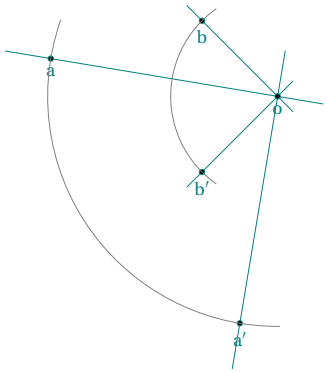
5.2.1 Example: method polar

```
\begin{tkzelements}
   scale   = .5
   z.O     = point:   new  (0, 0)
   z.A     = point:   new  (3, 0)
   z.F     = point:   polar (3, math.pi/6)
\end{tkzelements}
   \begin{tikzpicture}
      \tkzGetNodes
      \tkzDrawCircle(O,A)
      \tkzDrawSegments[new](O,A)
      \tkzDrawSegments[purple](O,F)
      \tkzDrawPoints(A,O,F)
      \tkzLabelPoints(A,O,F)
   \end{tikzpicture}
```



5.2.2 Example: rotation of points method set_rotation

```
\begin{tkzelements}
    z.a        = point:  new(0, -1)
    z.b        = point:  new(4, 0)
    z.o        = point:  new(6, -2)
    z.ap,z.bp  =  z.o : set_rotation (math.pi/2,z.a,z.b)
\end{tkzelements}
\begin{tikzpicture}
    \tkzGetNodes
    \tkzDrawLines(o,a o,a' o,b o,b')
    \tkzDrawPoints(a,a',b,b',o)
    \tkzLabelPoints(a,a',b,b',o)
    \tkzDrawArc(o,a)(a')
    \tkzDrawArc(o,b)(b')
 \end{tikzpicture}
```



6 Class line

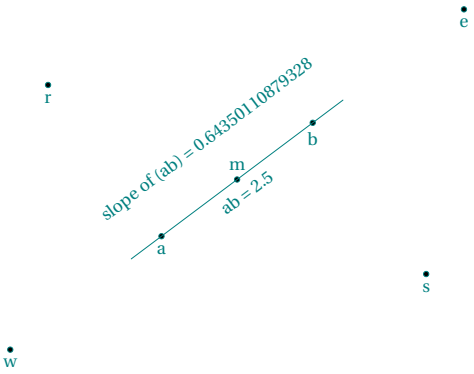
6.1 Attributs of a line

Writing `L.AB = line: new (z.A,z.B)` creates an object of the class **line** (the notation is arbitrary for the moment). Geometrically it is as much the line passing through the points A and B as the segment [AB]. Thus we can use the midpoint of `L.AB` which is of course the midpoint of the segment [AB]. This medium is obtained with `L.AB.mid`. Note that `L.AB.pa = z.A` and `L.AB.pb = z.B`. Finally, if a line L is the result of a method, you can obtain the points with `z.A,z.B = get_points (L)` or with the previous remark.
The attributes are :

Attributes	Application
pa	First point of the segment
pb	Second point of the segment
type	Type is 'line'
mid	Middle of the segment `z.M = L.AB.mid`
slope	Slope of the line obtained with `an = L.AB.slope`
length	Length of the segment `l = L.AB.length`
north_pa	See next example
north_pb	
south_pa	
south_pb	
east	
west	

6.1.1 Example: attributes of class line

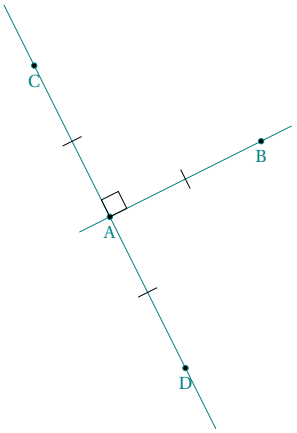
```
\begin{tkzelements}
  scale  = .5
   z.a   = point: new (1, 1)
   z.b   = point: new (5, 4)
   L.ab  = line : new (z.a,z.b)
   z.m   = L.ab.mid
   z.w   = L.ab.west
   z.e   = L.ab.east
   z.r   = L.ab.north_pa
   z.s   = L.ab.south_pb
   sl    = L.ab.slope
   len   = L.ab.length
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPoints(a,b,m,e,r,s,w)
   \tkzLabelPoints(a,b,e,r,s,w)
   \tkzLabelPoints[above](m)
   \tkzDrawLine(a,b)
   \tkzLabelSegment[sloped](a,b){ab = \tkzUseLua{len}}
   \tkzLabelSegment[above=12pt,sloped](a,b){slope of (ab) = \tkzUseLua{sl}}
\end{tikzpicture}
```



6.1.2 Method new and line attributes

Notation L or L.AB or L.euler. The notation is actually free. L.AB can also represent the segment. With L.AB = line : new a line is defined.

```
\begin{tkzelements}
   z.A   = point : new (1,1)
   z.B   = point : new (3,2)
   L.AB  = line : new (z.A,z.B)
   z.C   = L.AB.north_pa
   z.D   = L.AB.south_pa
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines(A,B C,D)
   \tkzDrawPoints(A,...,D)
   \tkzLabelPoints(A,...,D)
   \tkzMarkRightAngle(B,A,C)
   \tkzMarkSegments(A,C A,B A,D)
\end{tikzpicture}
```



6.2 Methods of the class line

Here's the list of methods for the **line** object. The results are either reals, points, lines, circles or triangles.

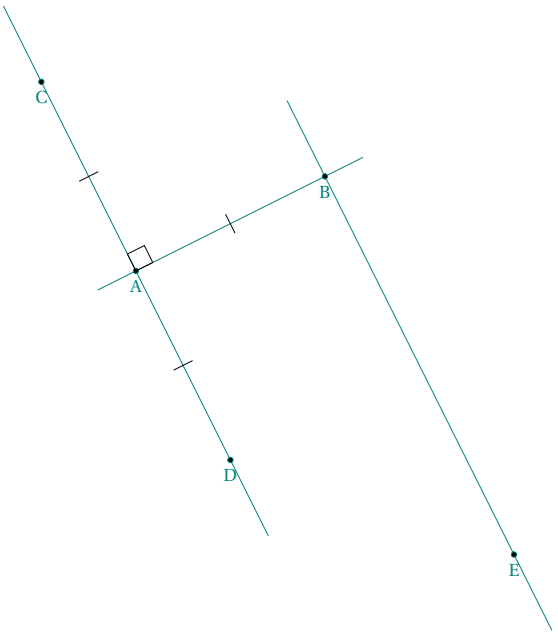
6.2.1 Table of the methods from class line

Methods	Comments
new(A, B)	L.AB = line : new(z.A,z.B) line through the points A and B
Points	
gold_ratio ()	z.C = L.AB : gold_ratio() –> gold ratio
normalize ()	z.C = L.AB : normalize() –> AC =1 and C ∈ (AB)
normalize_inv ()	z.C = L.AB : normalize_inv() –> CB =1 and C ∈ (AB)
barycenter (ka,kb)	z.C = L.AB : barycenter (1,2) C –> barycenter of {(A,1)(B,2)}
point (t)	z.C = L.AB : point (2) –> $\overrightarrow{AC} = 2\overrightarrow{AB}$
midpoint ()	z.M = L.AB : midpoint () –> better is z.M = L.AB.mid
harmonic_int	z.D = L.AB : harmonic_int (z.C) –> D ∈ [AB] C ∉ [AB]
harmonic_ext (pt)	z.D = L.AB : harmonic_ext (z.C) –> D ∉ [AB] C ∈ [AB]
harmonic_both (k)	z.C,z.D = L.AB : harmonic_both (tkzphi) –> CA/CB = DA/DB = tφ.
projection (pt)	z.H = L.AB : projection (z.C) –> CH ⊥ (AB) and H ∈ (AB)
set_projection (...)	projection of a list of points
symmetry_axial (pt)	z.Cp = L.AB : symmetry_axial (z.C)
set_symmetry_axial (...)	symmetry_axial of a list of points
translation (pt)	z.Cp = L.AB : translation (z.C)
set_translation(...)	translation of a list of points
square ()	z.C,z.D = L.AB : square () –> creates 2 points to make a square.
Lines	
ll_from (pt)	L.CD = L.AB : ll_from (z.C) –> (CD) ∥ (AB)
ortho_from (pt)	L.CD = L.AB : ortho_from (z.C) –> (CD) ⊥ (AB)
mediator ()	L.uv = L.AB : mediator () –> (u, v) mediator of (A, B)
Triangles	
equilateral ()	z.C = L.AB : equilateral_tr () $(\overrightarrow{AB},\overrightarrow{AC}) > 0$
isosceles (phi)	z.C = L.AB : isosceles (math.pi/6)
gold ()	z.C = L.AB : gold () –> right in B and AC = $\varphi \times$ AB
euclide ()	z.C = L.AB : euclide () –> AB = AC and $(\overrightarrow{AB},\overrightarrow{AC})$ = math.pi/5
golden ()	z.C = L.AB : golden () $(\overrightarrow{AB},\overrightarrow{AC}) = 2 \times$ math.pi/5
Circles	
circle ()	C.AB = L.AB : circle () circle center pa through pb
circle_swap ()	C.BA = L.AB : circle_swap () circle center pb through pa
Miscellaneous	
distance (pt)	d = L.Ab : distance (z.C) d=CH with H projection of C onto (AB)
in_out (pt)	b = L.AB : in_out (z.C) b is a boolean b=true if C ∈ (AB)
slope ()	a = L.AB : slope() –> better is L.AB.slope

Here are a few examples.

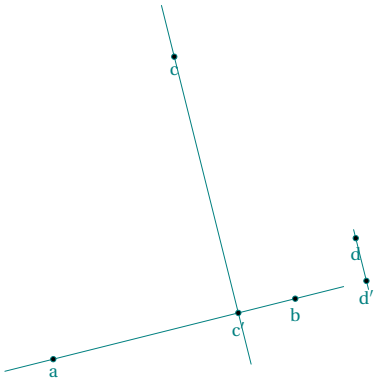
6.2.2 Example: new line from a defined line

```
\begin{tkzelements}
   scale = 1.25
   z.A   = point : new (1,1)
   z.B   = point : new (3,2)
   L.AB  = line : new (z.A,z.B)
   z.C   = L.AB.north_pa
   z.D   = L.AB.south_pa
   L.CD  = line : new (z.C,z.D)
   _,z.E   = get_points ( L.CD: ll_from (z.B))
   -- z.E   = L2.pb
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines(A,B C,D B,E)
   \tkzDrawPoints(A,...,E)
   \tkzLabelPoints(A,...,E)
   \tkzMarkRightAngle(B,A,C)
   \tkzMarkSegments(A,C A,B A,D)
\end{tikzpicture}
```



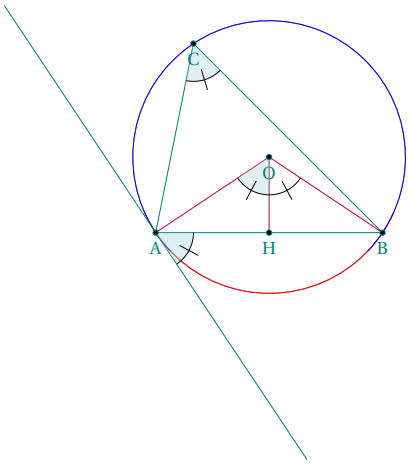
6.2.3 Example: projection of several points

```
\begin{tkzelements}
   scale     = .8
   z.a       = point:  new (0, 0)
   z.b       = point:  new (4, 1)
   z.c       = point:  new (2, 5)
   z.d       = point:  new (5, 2)
   L.ab      = line:   new (z.a,z.b)
   z.cp,z.dp = L.ab:  set_projection(z.c,z.d)
\end{tkzelements}
 \begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines(a,b c,c' d,d')
   \tkzDrawPoints(a,...,d,c',d')
   \tkzLabelPoints(a,...,d,c',d')
 \end{tikzpicture}
```



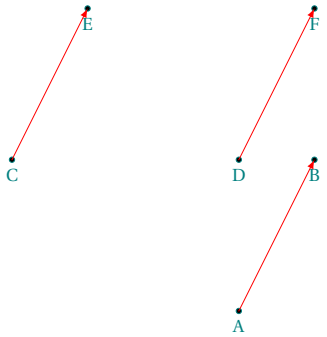
6.2.4 Example: combination of methods

```
\begin{tkzelements}
   z.A     = point: new (0 , 0)
   z.B     = point: new (6 , 0)
   z.C     = point: new (1 , 5)
   T.ABC   = triangle: new (z.A,z.B,z.C)
   L.AB    = T.ABC.ab
   z.O     = T.ABC.circumcenter
   C.OA    = circle: new (z.O,z.A)
   z.H     = L.AB: projection (z.O)
   L.ab    = C.OA: tangent_at (z.A)
   z.a,z.b = L.ab.pa,L.ab.pb
  -- or z.a,z.b  = get_points (L.ab)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(A,B,C)
   \tkzDrawCircle(O,A)
   \tkzDrawSegments[purple](O,A O,B O,H)
   \tkzDrawArc[red](O,A)(B)
   \tkzDrawArc[blue](O,B)(A)
   \tkzDrawLine[add = 2 and 1](A,a)
   \tkzFillAngles[teal!30,opacity=.4](A,C,B b,A,B A,O,H)
   \tkzMarkAngles[mark=|](A,C,B b,A,B A,O,H H,O,B)
   \tkzDrawPoints(A,B,C,H,O)
   \tkzLabelPoints(A,B,C,H,O)
\end{tikzpicture}
```



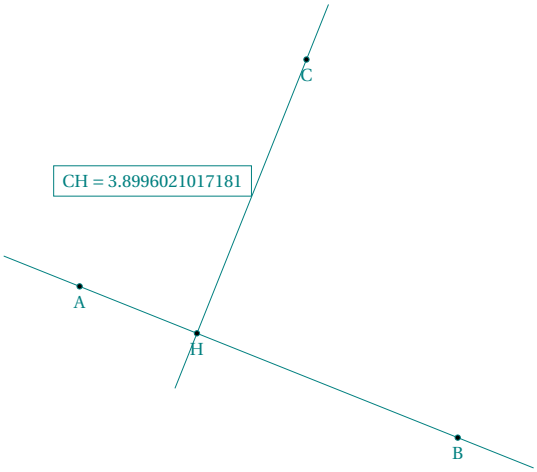
6.2.5 Example: translation

```
\begin{tkzelements}
   z.A  = point:  new (0,0)
   z.B  = point:  new (1,2)
   z.C  = point:  new (-3,2)
   z.D  = point:  new (0,2)
   L.AB = line : new (z.A,z.B)
 -- z.E = L.AB : translation (z.C)
   z.E,z.F = L.AB : set_translation (z.C,z.D)
\end{tkzelements}
\begin{tikzpicture}
\tkzGetNodes
\tkzDrawPoints(A,...,F)
\tkzLabelPoints(A,...,F)
\tkzDrawSegments[->,red,> =latex](C,E D,F A,B) )
\end{tikzpicture}
```



6.2.6 Example: distance and projection

```
\begin{tkzelements}
   z.A    = point : new (0 , 0)
   z.B    = point : new (5 , -2)
   z.C    = point : new (3 , 3)
   L.AB   = line : new (z.A,z.B)
   d      = L.AB : distance (z.C)
   z.H    = L.AB : projection (z.C)
\end{tkzelements}
\begin{tikzpicture}
  \tkzGetNodes
  \tkzDrawLines(A,B C,H)
  \tkzDrawPoints(A,B,C,H)
  \tkzLabelPoints(A,B,C,H)
  \tkzLabelSegment[above left,
                   draw](C,H){$CH = \tkzUseLua{d}$}
\end{tikzpicture}
```



7 Class circle

7.1 Attributs of a circle

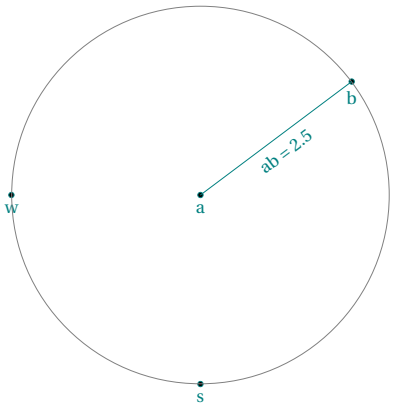
This class is also defined by two points, on the one hand the center and on the other hand a point through which the circle passes.

Attributes	Application
center	z.A = C.AB.center
through	z.B = C.AB.through
type	C.AB.type -> C.OA.type = circle
radius	C.AB.radius -> r = C.OA.radius r real number
north	C.AB.north -> z.N = C.OA.north
south	C.AB.south -> z.S = C.OA.south
east	C.AB.east -> z.E = C.OA.east
west	C.AB.west -> z.W = C.OA.west

7.1.1 Example: circle attributes

Three attributes are used (south, west, radius).

```
\begin{tkzelements}
   scale = .5
   z.a   = point: new (1, 1)
   z.b   = point: new (5, 4)
   C.ab  = circle : new (z.a,z.b)
   z.s   = C.ab.south
   z.w   = C.ab.west
   r     = C.ab.radius
\end{tkzelements}
\begin{tikzpicture}
    \tkzGetNodes
    \tkzDrawPoints(a,b,s,w)
    \tkzLabelPoints(a,b,s,w)
    \tkzDrawCircle(a,b)
    \tkzDrawSegment(a,b)
    \tkzLabelSegment[sloped](a,b){ab = \tkzUseLua{r}}
\end{tikzpicture}
```



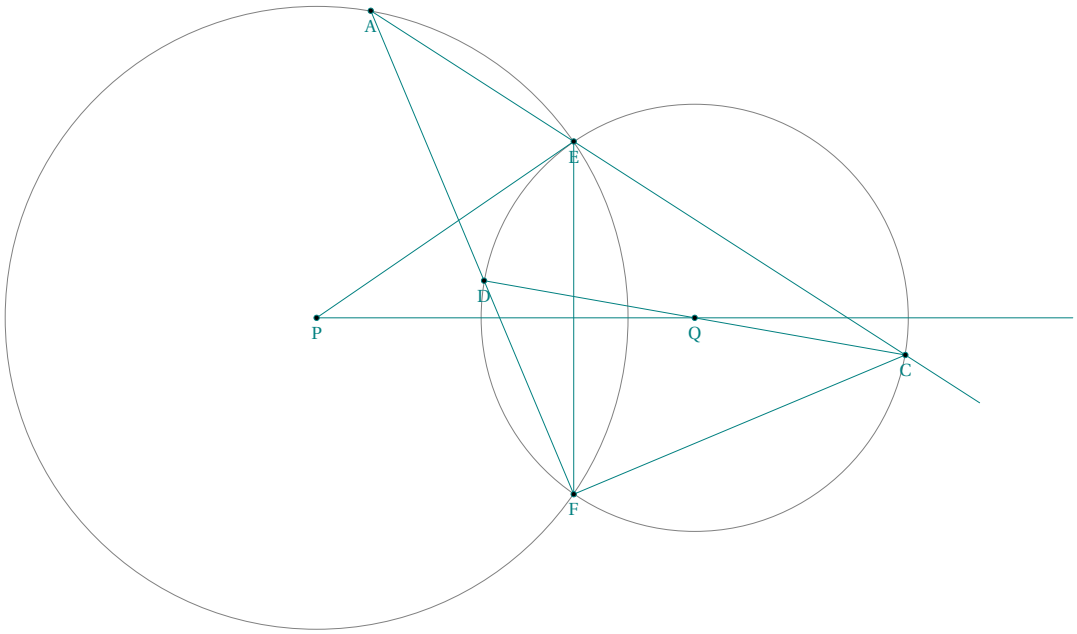
7.2 Methods of the class circle

Methods	Comments
new(O,A)	C.OA = circle : new (z.O,z.A) –> circle center O through A
radius(O,r)	C.OA = circle : radius (z.O,2) –> circle center O radius =2 cm
Points	
antipode (pt)	z.C = C.OA: antipode (z.B) –> [BC] is a dimater
inversion (pt)	z.Bp = C.AC:inversion (z.B)
set_inversion (list of pts)	z.Bp,z.Ep,z.Cp = C.AC: set_inversion (z.B, z.E, z.C)
midarc (z1,z2)	z.D = C.AB: midarc (z.B,z.C) D is the midarc of $\overset{\frown}{AB}$
point (phi)	z.E = C.AB: point (-math.pi/3)
random_pt(lower, upper)	
internal_similitude (C)	z.I = C.1 : internal_similitude (C.2)
external_similitude (C)	z.J = C.1 : external_similitude (C.2)
Lines	
radical_axis (C)	
tangent_at (pt)	z.P = C.OA: tangent_at (z.M) ((PM) $\perp$ (OM))
tangent_from (pt)	z.M,z.N = C.OA: tangent_from (z.P)
inversion (line)	L or C = C.AC:inversion (L.EF)
Circles	
orthogonal_from (pt)	C= C.OA: orthogonal_from (z.P)
orthogonal_through (pta,ptb)	C = C.OA: orthogonal_through (z.z1,z.z2)
inversion (circle)	L or C = C.AC:inversion (C.EF)
Miscellaneous	
power (pt)	p = C.OA: power (z.M) –> power with respect to a circle
in_out (pt)	C.OA : in_out (z.M) –> boolean
in_out_disk (pt)	C.OA : in_out_disk (z.M) –> boolean
draw ()	for further use

7.2.1 Altshiller

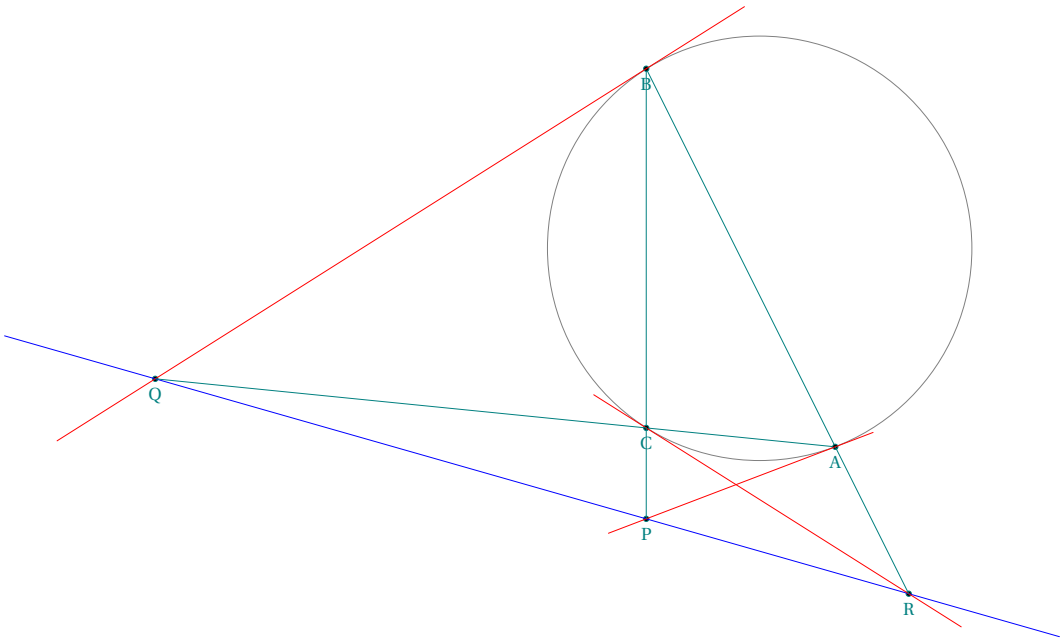
```
\begin{tkzelements}
   z.P  = point : new (0,0)
   z.Q  = point : new (5,0)
   z.I  = point : new (3,2)
   C.QI = circle :    new (z.Q,z.I)
   C.PE = C.QI : orthogonal_from (z.P)
   z.E  = C.PE.through
   C.QE = circle :    new (z.Q,z.E)
   _,z.F    = intersection (C.PE,C.QE)
   z.A  = C.PE: point (math.pi/180*80)
   L.AE = line : new (z.A,z.E)
   _,z.C    = intersection (L.AE,C.QE)
   L.AF = line : new (z.A,z.F)
   L.CQ = line : new (z.C,z.Q)
   z.D  = intersection (L.AF,L.CQ)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(P,E Q,E)
   \tkzDrawLines[add=0 and 1](P,Q)
```

```
   \tkzDrawLines[add=0 and 2](A,E)
   \tkzDrawSegments(P,E E,F F,C A,F C,D)
   \tkzDrawPoints(P,Q,E,F,A,C,D)
   \tkzLabelPoints(P,Q,E,F,A,C,D)
\end{tikzpicture}
```



7.2.2 Lemoine

```
\begin{tkzelements}
   scale = 1.6
   z.A   = point: new (1,0)
   z.B   = point: new (5,2)
   z.C   = point: new (1.2,2)
   T     = triangle: new(z.A,z.B,z.C)
   z.O   = T.circumcenter
   C.OA  = circle: new (z.O,z.A)
   L.tA  = C.OA: tangent_at (z.A)
   L.tB  = C.OA: tangent_at (z.B)
   L.tC  = C.OA: tangent_at (z.C)
   z.P   = intersection (L.tA,T.bc)
   z.Q   = intersection (L.tB,T.ac)
   z.R   = intersection (L.tC,T.ab)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon[teal](A,B,C)
   \tkzDrawCircle(O,A)
   \tkzDrawPoints(A,B,C,P,Q,R)
   \tkzLabelPoints(A,B,C,P,Q,R)
   \tkzDrawLine[blue](Q,R)
   \tkzDrawLines[red](A,P B,Q R,C)
   \tkzDrawSegments(A,R C,P C,Q)
\end{tikzpicture}
```



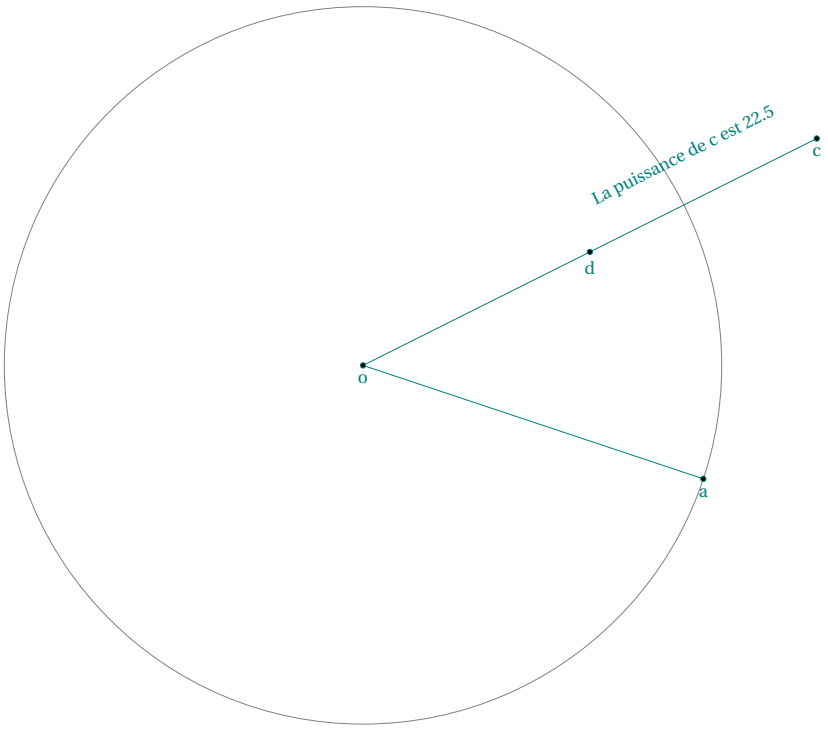
7.2.3 Inversion: point, line and circle

The "inversion" method can be used on a point, a line or a circle. Depending on the type of object, the function determines the correct algorithm to use.

7.2.4 Inversion: point

The "inversion" method can be used on a point, a line or a circle. Depending on the type of object, the function determines the correct algorithm to use.

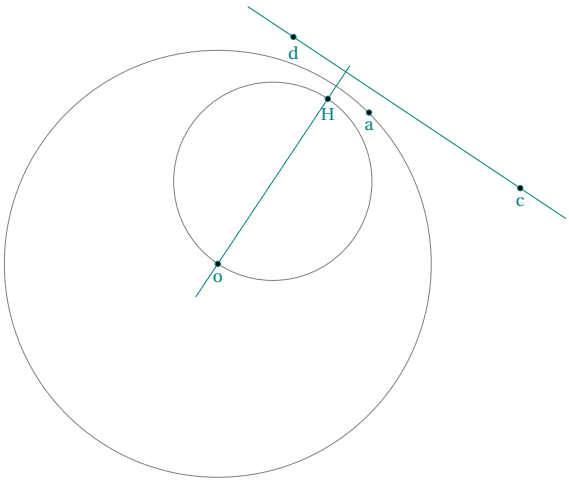
```
\begin{tkzelements}
   scale = 1.5
   z.o   = point:    new (-1,2)
   z.a   = point:    new (2,1)
   C.oa  = circle:   new (z.o,z.a)
   z.c   = point:    new (3,4)
   z.d   = C.oa:     inversion (z.c)
   p     = C.oa:     power (z.c)
\end{tkzelements}
\begin{tikzpicture}
    \tkzGetNodes
    \tkzDrawCircle(o,a)
    \tkzDrawSegments(o,a o,c)
    \tkzDrawPoints(a,o,c,d)
    \tkzLabelPoints(a,o,c,d)
    \tkzLabelSegment[sloped,above=1em](c,d){La puissance de c est \tkzUseLua{p}}
 \end{tikzpicture}
```



7.2.5 Inversion: line

The result is either a straight line or a circle.

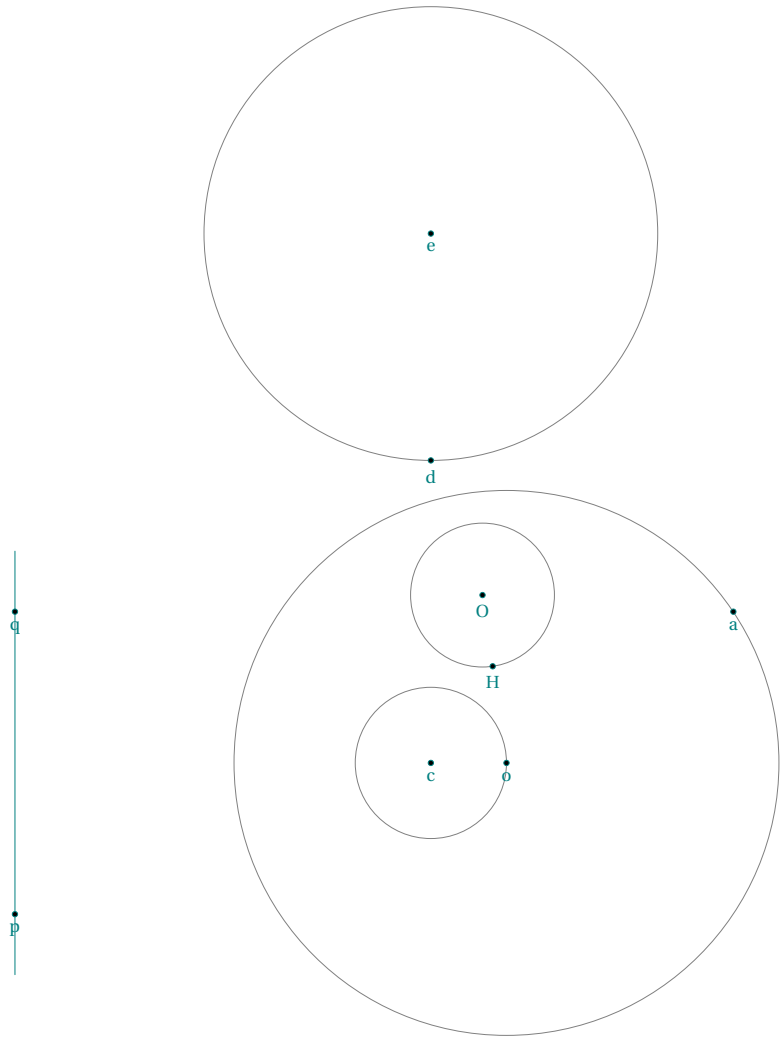
```
\begin{tkzelements}
   scale    = 1
   z.o      = point:    new (-1,1)
   z.a      = point:    new (1,3)
   C.oa     = circle:   new (z.o,z.a)
   z.c      = point:    new (3,2)
   z.d      = point:    new (0,4)
   L.cd     = line:    new (z.c,z.d)
   C.OH     = C.oa: inversion (L.cd)
   z.O,z.H  = get_points(C.OH)
\end{tkzelements}
\begin{tikzpicture}
    \tkzGetNodes
    \tkzDrawCircles(o,a O,H)
    \tkzDrawLines(c,d o,H)
    \tkzDrawPoints(a,o,c,d,H)
    \tkzLabelPoints(a,o,c,d,H)
 \end{tikzpicture}
```



7.2.6 Inversion: circle

The result is either a straight line or a circle.

```
\begin{tkzelements}
   scale = 1
   z.o   = point:    new (-1,1)
   z.a   = point:    new (2,3)
   C.oa  = circle:   new (z.o,z.a)
   z.c   = point:    new (-2,1)
   z.e   = point:    new (-2,8)
    z.d  = point:    new (-2,5)
   C.ed  = circle:   new (z.e,z.d)
   C.co  = circle:   new (z.c,z.o)
   obj   = C.oa: inversion (C.co)
   if obj.type == "line"
      then
          z.p,z.q = get_points(obj)
      else
         z.0,z.H = get_points(obj)
   end
    obj   = C.oa: inversion(C.ed)
   if obj.type == "line"
      then
          z.p,z.q = get_points(obj)
      else
         z.0,z.H = get_points(obj)
   end
\end{tkzelements}
\begin{tikzpicture}
    \tkzGetNodes
    \tkzDrawCircles(o,a c,o e,d 0,H)
    \tkzDrawLines(p,q)
    \tkzDrawPoints(a,o,c,d,e,p,q,0,H)
   \tkzLabelPoints(a,o,c,d,e,p,q,0,H)
 \end{tikzpicture}
```


e
d
O
a
H
q
c
o
p

8 Classe triangle

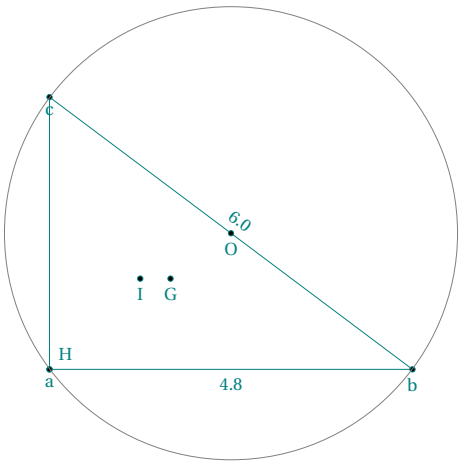
8.1 Attributs of a triangle

The triangle object is created using the new method, for example with `T.abc = triangle : new (z.a,z.b,z.c)` (See examples: 12.3; 12.4; 12.11). Multiple attributes are then created.

Attributes	Application
pa	
pb	
pc	
circumcenter	
centroid	
incenter	
orthocenter	
eulercenter	
a	It's the length of the side opposite the first vertex
b	It's the length of the side opposite the second verte
c	It's the length of the side opposite the third vertex
alpha	Vertex angle of the first vertex
beta	Vertex angle of the second vertex
gamma	Vertex angle of the third vertex
ab	Line defined by the first two points of the triangle
ac	Line defined by the first point and the last point
bc	Line defined by the last two points of the triangle

8.1.1 Example: triangle attributes

```
\begin{tkzelements}
   z.a   = point: new (0 , 0)
   z.b   = point: new (4 , 0)
   z.c   = point: new (0 , 3)
   T.abc = triangle : new (z.a,z.b,z.c)
   z.O   = T.abc.circumcenter
   z.I   = T.abc.incenter
   z.H   = T.abc.orthocenter
   z.G   = T.abc.centroid
   a     = T.abc.a
   b     = T.abc.b
   c     = T.abc.c
   alpha =  T.abc.alpha
   beta  =  T.abc.beta
   gamma =  T.abc.gamma
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(a,b,c)
   \tkzDrawPoints(a,b,c,O,G,I,H)
   \tkzLabelPoints(a,b,c,O,G,I)
   \tkzLabelPoints[above right](H)
   \tkzDrawCircles(O,a)
   \tkzLabelSegment[sloped](a,b){\tkzUseLua{c}}
   \tkzLabelSegment[sloped,above](b,c){\tkzUseLua{a}}
\end{tikzpicture}
```



8.2 Methods of the class triangle

Methods	Comments
triangle = class triangle	
new (a, b ,c)	T.ABC = triangle : new (z.A,z.B,z.C)
...	T or T.name with what you want for name, is possible.
Points	
lemoine_point ()	T.ABC : lemoine_point () intersection os the symmedians
symmedian_point ()	Lemoine point or the Grebe point
bevan_point ()	Circumcenter of the excentral triangle
mittenpunkt_point ()	Symmedian point of the excentral triangle
gergonne_point ()	Intersection of the three cevians that lead to the contact points
nagel_point ()	Intersection of the three cevians that lead to the extouch points
feuerbach_point ()	The point at which the incircle and euler circle are tangent.
spieker_center ()	Incenter of the medial triangle
barycenter (ka,kb,kc)	T.ABC: barycenter (2,1,1) barycenter of ({A,2},{B,1},{C,1})
base (u,v)	z.D = T.ABC: base(1,1) -> ABDC is a parallelogram
projection (p)	Projection of a point on the sides
euler_points ()	Euler points of euler circle
nine_points ()	Points of the euler circle
parallelogram ()	z.D = T.ABC : parallelogram () -> ABCD is a parallelogram
Lines	
altitude (n)	L.AHa = T.ABC : altitude () n empty or 0 line from A [a]
bisector (n)	L.Bb = T.ABC : bisector (1) n = 1 line from B [b]
bisector_ext(n)	n=2 line from the third vertex.
symmedian_line (n)	Cevian with respect to Lemoine point.
euler_line ()	the line through N ,G, H and O if the triangle is not equilateral [c]
antiparallel(pt,n)	n=0 antiparallel through pt to (BC), n=1 to (AC) etc.

a z.Ha = L.AHa.pb recovers the common point of the opposite side and altitude. The method orthic is usefull.
b _,z.b = get_points(L.Bb) recovers the common point of the opposite side and bisector.
c N center of nine points circle, G centroid, H orthocenter , O circum center

Methods	Comments
Circles	
euler_circle ()	C.NP = T.ABC : euler_circle () -> N euler point [a]
circum_circle ()	C.OA = T.ABC : circum () Triangle's circumscribed circle
in_circle ()	Inscribed circle of the triangle
ex_circle (n)	Circle tangent to three lines defined by the sides of the triangle
first_lemoine_circle ()	The center is the midpoint between Lemoine point and the circumcenter.[b]
second_lemoine_circle ()	see example 12.63
spieker_circle ()	The incircle of the medial triangle
ex_circle(n)	z.o,z.p,z.q,z.r = T.ABC: ex_circle ()
...	n=0 ex_circle opposed to A p,q,r projections of o on the sides
Triangles	
orthic ()	T = T.ABC : orthic () triangle joining the feet of the altitudes
medial ()	T = T.ABC : medial () triangle with vertices at the midpoints
incentral ()	Cevian triangle of the triangle with respect to its incenter
excentral ()	Triangle with vertices corresponding to the excenters
extouch ()	Triangle formed by the points of tangency with the excircles
intouch ()	Contact triangle formed by the points of tangency of the incircle
tangential ()	Triangle formed by the lines tangent to the circumcircle at the vertices
feuerbach ()	Triangle formed by the points of tangency of the euler circle with the excircles
anti ()	Anticomplementary Triangle The given triangle is its medial triangle.
cevian (pt)	Triangle formed with the endpoints of the three cevians with respect to pt.
symmedian ()	Triangle formed with the intersection points of the symmedians.
euler ()	Triangle formed with the euler points
Miscellaneous	
area ()	$\mathscr{A}$ = T.ABC: area ()
barycentric_coordinates (pt)	Triples of numbers corresponding to masses placed at the vertices
in_out (pt)	Boolean. Test if pt is inside the triangle
check_equilateral ()	Boolean. Test if the triangle is equilateral

a The midpoint of each side of the triangle, the foot of each altitude, the midpoint of the line segment from each vertex of the triangle to the orthocenter.

b Through the Lemoine point draw lines parallel to the triangle's sides. The points where the parallel lines intersect the sides of ABC then lie on a circle known as the first Lemoine circle.

9 Classe ellipse

9.1 Attributs of an ellipse

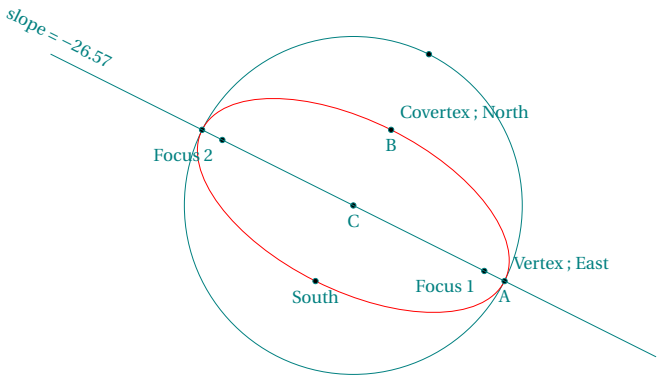
The first attributes are the three points that define the ellipse : `center` , `vertex` and `covertex`. The first method to define an ellipse is to give its center, then the point named **`vertex`** which defines the major axis and finally the point named **`covertex`** which defines the minor axis.

Attributes	Application
`center`	center of the ellipse
`vertex`	point of the major axis and of the ellipse
`covertex`	point of the minor axis and of the ellipse
`type`	The type is 'ellipse'
`Rx`	Radius from center to vertex
`Ry`	Radius from center to covertex
`slope`	Slope of the line passes through the foci
`Fa`	First focus
`Fb`	Second focus
`south`	See next example 9.1.1
`north`	See next example 9.1.1
`west`	See next example 9.1.1
`east`	See next example 9.1.1

9.1.1 Attributs of an ellipse

```
\begin{tkzelements}
   z.C   = point: new (3 , 2)
   z.A   = point: new (5 , 1)
   L.CA  = line : new (z.C,z.A)
   z.b   = L.CA.north_pa
   L     = line : new (z.C,z.b)
   z.B   = L : point (0.5)
   E     = ellipse: new (z.C,z.A,z.B)
   a     = E.Rx
   b     = E.Ry
   z.F1  = E.Fa
   z.F2  = E.Fb
   slope = math.deg(E.slope)
   z.E   = E.east
   z.N   = E.north
   z.W   = E.west
   z.S   = E.south
   z.Co  = E.covertex
   z.Ve  = E.vertex
\end{tkzelements}
\begin{tikzpicture}
   \pgfkeys{/pgf/number format/.cd,fixed,precision=2}
   \tkzGetNodes
   \tkzDrawCircles[teal](C,A)
   \tkzDrawEllipse[red](C,\tkzUseLua{a},\tkzUseLua{b},\tkzUseLua{slope})
   \tkzDrawPoints(C,A,B,b,W,S,F1,F2)
   \tkzLabelPoints(C,A,B)
   \tkzDrawLine[add = .5 and .5](A,W)
   \tkzLabelSegment[pos=1.5,above,sloped](A,W){slope = \pgfmathprintnumber{\slope}}
   \tkzLabelPoint[below](S){South}
   \tkzLabelPoint[below left](F1){Focus 1}
```

```
   \tkzLabelPoint[below left](F2){Focus 2}
   \tkzLabelPoint[above right](Ve){Vertex ; East}
   \tkzLabelPoint[above right](Co){Covertex ; North}
\end{tikzpicture}
```



9.2 Methods of the class ellipse

Before reviewing the methods and functions related to ellipses, let's take a look at how you can draw ellipses with tkz-elements. The \tkzDrawEllipse macro requires 4 arguments: the center of the ellipse, the long radius (on the focus axis), the short radius and the angle formed by the focus axis. The last three arguments must be transferred from **tkzelements** to **tikzpicture**. To do this, you'll need to use a tkz-elements function: set_lua_to_tex. See 10 or the next examples.

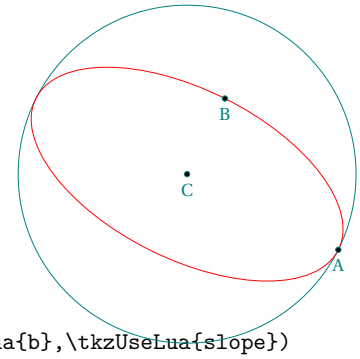
☞ You need to proceed with care, because unfortunately at the moment, the macros you create are global and you can overwrite existing macros. One solution is either to choose a macro name that won't cause any problems, or to save the initial macro.

Methods	**Example**
new (pc, pa ,pb)	E = ellipse: new (center, vertex, covertex)
foci (f1,f2,v)	E = ellipse: foci (focus 1, focus 2, vertex)
radii (c,a,b,sl)	E = ellipse: radii (center, radius a, radius b, slope)
in_out (pt)	pt in/out of the ellipse
tangent_at (pt)	see example 9.2.3
tangent_from (pt)	see example 9.2.3
point (phi)	vertex = point (0) covertex = point (math.pi/4) ex see 9.2.3

9.2.1 Method new

The main method for creating a new ellipse is new. The arguments are three: center, vertex and covertex For attributes see 9

```
\begin{tkzelements}
   z.C      = point: new (3 , 2)
   z.A      = point: new (5 , 1)
   z.B      = z.C : homothety(0.5,
              z.C : rotation (math.pi/2,z.A))
   E        = ellipse: new (z.C,z.A,z.B)
   a        = E.Rx
   b        = E.Ry
   slope    = math.deg(E.S1)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles[teal](C,A)
   \tkzDrawEllipse[red](C,\tkzUseLua{a},\tkzUseLua{b},\tkzUseLua{slope})
   \tkzDrawPoints(C,A,B)
   \tkzLabelPoints(C,A,B)
\end{tikzpicture}
```



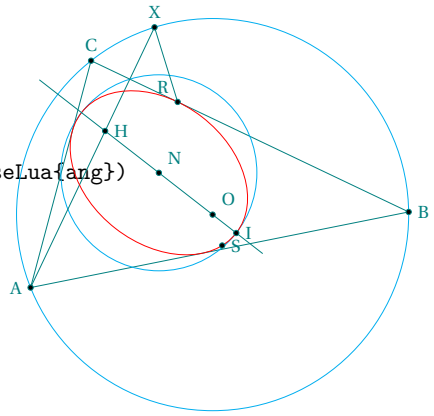
The function `set_lua_to_tex (list)` is used to define the macros that will be used to draw the ellipse with Ti*kZ* or `tkz-euclide`.

9.2.2 Method foci

The first two points are the foci of the ellipse. The third one is the vertex. We can deduce all the other characteristics.
The function launches the `new` *method, all the characteristics of the ellipse are defined.*

```
\begin{tkzelements}
   z.A      = point: new (0 , 0)
   z.B      = point: new (5 , 1)
   L.AB     = line : new (z.A,z.B)
   z.C      = point: new (.8 , 3)
   T.ABC    = triangle: new (z.A,z.B,z.C)
   z.N      = T.ABC.eulercenter
   z.H      = T.ABC.orthocenter
   z.O      = T.ABC.circumcenter
   _,_,z.Mc = get_points (T.ABC: medial ())
   L.euler  = line: new (z.H,z.O)
   C.circum = circle: new (z.O,z.A)
   C.euler  = circle: new (z.N,z.Mc)
   z.i,z.j  = intersection (L.euler,C.circum)
   z.I,z.J  = intersection (L.euler,C.euler)
   E        = ellipse: foci (z.H,z.O,z.I)
   L.AH     = line: new (z.A,z.H)
   z.X      = intersection (L.AH,C.circum)
   L.XO     = line: new (z.X,z.O)
   z.R,z.S  = intersection (L.XO,E)
   a,b      = E.Rx,E.Ry
   ang      = math.deg(E.slope)
\end{tkzelements}
```

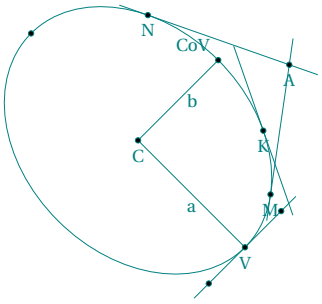
```
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(A,B,C)
   \tkzDrawCircles[cyan](O,A N,I)
   \tkzDrawSegments(X,R A,X)
   \tkzDrawEllipse[red](N,\tkzUseLua{a},\tkzUseLua{b},\tkzUseLua{ang})
   \tkzDrawLines[add=.2 and .5](I,H)
   \tkzDrawPoints(A,B,C,N,O,X,H,R,S,I)
   \tkzLabelPoints[above](C,X)
   \tkzLabelPoints[above right](N,O)
   \tkzLabelPoints[above left](R)
   \tkzLabelPoints[left](A)
   \tkzLabelPoints[right](B,I,S,H)
   \end{tikzpicture}
```



9.2.3 Method point and radii

The method `point` defines a point M of the ellipse whose coordinates are (a × cos(phi), b × sin(phi)). `phi` angle between (center,vertex) and (center,M)
The environment ***tkzelements*** *uses as* **lua** *the radian as unit for angles.*

```
\begin{tkzelements}
   z.C          = point: new (2 , 3)
   z.A          = point: new (6 , 5)
   a            = value(4)
   b            = value(3)
   ang          = math.deg(-math.pi/4)
   E            = ellipse: radii (z.C,a,b,-math.pi/4)
   z.V          = E : point (0)
   z.K          = E : point (1)
   z.CoV        = E : point (math.pi/2)
   z.X          = E : point (math.pi)
   L            = E :tangent_at (z.V)
   z.x,z.y      = get_points(L)
   L.ta,L.tb    = E :tangent_from (z.A)
   z.M          = L.ta.pb
   z.N          = L.tb.pb
   L.K          = E :tangent_at (z.K)
   z.ka,z.kb    = get_points(L.K)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawSegments(C,V C,CoV)
   \tkzDrawLines(x,y A,M A,N ka,kb)
   \tkzLabelSegment(C,V){$a$}
   \tkzLabelSegment[right](C,CoV){$b$}
   \tkzDrawEllipse[teal](C,\tkzUseLua{a},\tkzUseLua{b},\tkzUseLua{ang})
   \tkzDrawPoints(C,V,CoV,X,x,y,M,N,A,K)
   \tkzLabelPoints(C,V,A,M,N,K)
   \tkzLabelPoints[above left](CoV)
\end{tikzpicture}
```



10 Math constants and functions

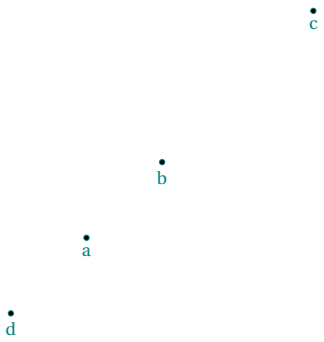
contants or functions	Comments
tkzphi	constant $\varphi = (1 + \text{math.sqrt}(5))/2$
tkzinvphi	constant $1/\varphi = 1/\text{tkzphi}$
tkzsqrtphi	constant $\sqrt{\varphi} = \text{math.sqrt}(\text{tkzphi})$
islinear (z1,z2,z3)	Are the points aligned? (z2-z1) ∥ (z3-z1) ?
isortho (z1,z2,z3)	(z2-z1) ⊥ (z3-z1) ? boolean
set_lua_to_tex (list)	set_lua_to_tex('a','n') defines \a and \n
tkzUseLua (variable)	\textbackslash\tkzUseLua{a} prints the value of a
value (v)	apply scale * value
real (v)	apply value /scale
angle_normalize (a)	to get a value between 0 and 2π
radical_center (C1,C2,C3)	see 12.35
radical_circle (C1,C2,C3)	see 12.36

10.0.1 Harmonic division with tkzphi

```
\begin{tkzelements}
   scale =.5
   z.a = point: new(0,0)
   z.b = point: new(8,0)
   L.ab = line: new (z.a,z.b)
   z.m,z.n = L.ab: harmonic_both (tkzphi)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLine[add= .2 and .2](a,n)
   \tkzDrawPoints(a,b,n,m)
   \tkzLabelPoints(a,b,n,m)
\end{tikzpicture}
```

10.0.2 Function islinear

```
\begin{tkzelements}
   z.a = point: new (1, 1)
   z.b = point: new (2, 2)
   z.c = point: new (4, 4)
   if islinear (z.a,z.b,z.c) then
       z.d = point: new (0, 0)
    else
        z.d = point: new (-1, -1)
   end
\end{tkzelements}
\begin{tikzpicture}
    \tkzGetNodes
    \tkzDrawPoints(a,...,d)
    \tkzLabelPoints(a,...,d)
\end{tikzpicture}
```



10.0.3 Function value

value to apply scaling if necessary

If scale = 1.2 with a = value(5) the actual value of a will be $5 \times 1.2 = 6$.

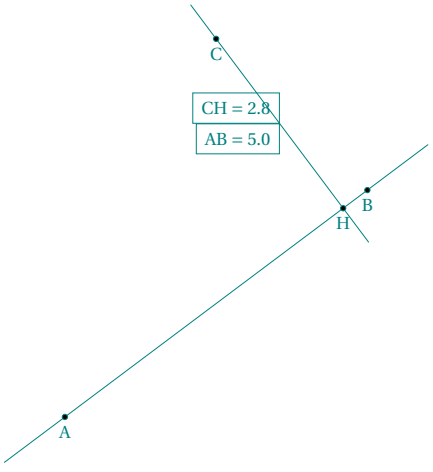
10.0.4 Function real

If scale = 1.2 with a = 6 then real(a) = 6/1.2 = 5 .

10.0.5 Transfer from lua to TEX

It's possible to transfer variable from Lua to TEX with \tkzUseLua in the environment "tikzpicture".

```
\begin{tkzelements}
   z.A            = point : new (0 , 0)
   z.B            = point : new (4 , 3)
   z.C            = point : new (2 , 5)
   L.AB           = line : new (z.A,z.B)
   d              = L.AB : distance (z.C)
   l              = L.AB.length
   z.H            = L.AB : projection (z.C)
\end{tkzelements}
\begin{tikzpicture}
\tkzGetNodes
\tkzDrawLines(A,B C,H)
\tkzDrawPoints(A,B,C,H)
\tkzLabelPoints(A,B,C,H)
\tkzLabelSegment[above left,draw](C,H){$CH = \tkzUseLua{d}$}
\tkzLabelSegment[below left,draw](C,H){$AB = \tkzUseLua{l}$}
\end{tikzpicture}
```

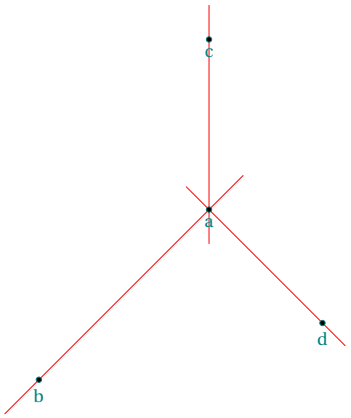


10.0.6 Normalized angles : Slope of lines (ab), (ac) and (ad)

```
\begin{tkzelements}
   z.a      = point: new(0, 0)
   z.b      = point: new(-3, -3)
   z.c      = point: new(0, 3)
   z.d      = point: new(2, -2)
   angle    = point.arg (z.b-z.a)
   tex.print('slope of (ab) : '..tostring(angle)..'\\\\')
   tex.print('slope normalized of (ab) : '..tostring(angle\_normalize(angle))..'\\\\')
   angle    = point.arg (z.c-z.a)
```

```
   tex.print('slope of (ac) : '..tostring(angle)..'\\\\')
   tex.print('slope normalized of (ac) : '..tostring(angle\_normalize(angle))..'\\\\')
   angle    = point.arg (z.d-z.a)
   tex.print('slope of (ad) : '..tostring(angle)..'\\\\')
   tex.print('slope normalized of (acd) : '..tostring(angle\_normalize(angle))..'\\\\')
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines[red](a,b a,c a,d)
   \tkzDrawPoints(a,b,c,d)
   \tkzLabelPoints(a,b,c,d)
\end{tikzpicture}
```

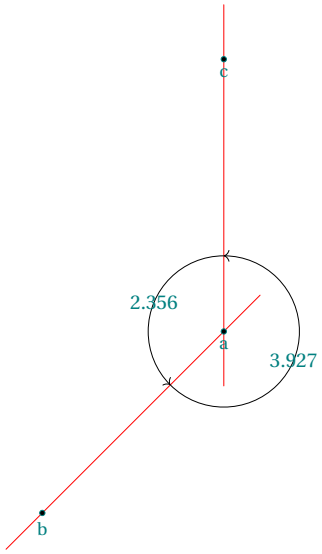
slope of (ab) : -2.3561944901923
slope normalized of (ab) : 3.9269908169872
slope of (ac) : 1.5707963267949
slope normalized of (ac) : 1.5707963267949
slope of (ad) : -0.78539816339745
slope normalized of (ad) : 5.4977871437821



10.0.7 Get angle

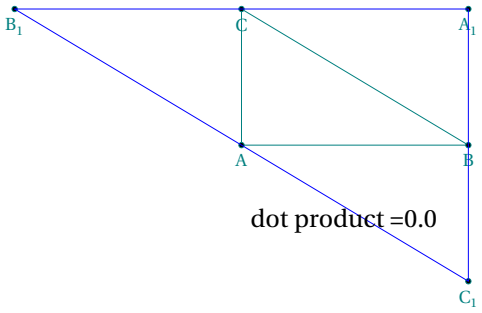
The function `get_angle (a,b,c)` gives the angle normalized of $(\overrightarrow{ab},\overrightarrow{ac})$.

```
\begin{tkzelements}
   z.a   = point: new(0, 0)
   z.b   = point: new(-2, -2)
   z.c   = point: new(0, 3)
   angcb = tkzround ( get_angle (z.a,z.c,z.b),3)
   angbc = tkzround ( get_angle (z.a,z.b,z.c),3)
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines[red](a,b a,c)
   \tkzDrawPoints(a,b,c)
   \tkzLabelPoints(a,b,c)
   \tkzMarkAngle[->](c,a,b)
   \tkzLabelAngle(c,a,b){\tkzUseLua{angcb}}
   \tkzMarkAngle[->](b,a,c)
   \tkzLabelAngle(b,a,c){\tkzUseLua{angbc}}
\end{tikzpicture}
```



10.0.8 Dot or scalar product

```
\begin{tkzelements}
   z.A    = point: new(0,0)
   z.B    = point: new(5,0)
   z.C    = point: new(0,3)
   T.ABC  = triangle: new (z.A,z.B,z.C)
   z.A_1,
   z.B_1,
   z.C_1  = get_points (T.ABC: anti ())
   x  = dot_product (z.A,z.B,z.C)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(A,B,C)
   \tkzDrawPoints(A,B,C,A_1,B_1,C_1)
   \tkzLabelPoints(A,B,C,A_1,B_1,C_1)
   \tkzDrawPolygon[blue](A_1,B_1,C_1)
   \tkzText[right](0,-
1){dot product =\tkzUseLua{x}}
\end{tikzpicture}
```



The scalar product of the vectors $\overrightarrow{AC}$ and $\overrightarrow{AB}$ is equal to 0.0, so these vectors are orthogonal.

10.0.9 Alignment or orthogonality

With the functions `islinear` and `isortho`. `islinear(z.a,z.b,z.c)` gives `true` idf the points a, b and c are aligned.
`isortho(z.a,z.b,z.c)` gives `true` if the line (ab) is orthogonal to the line (ac).

10.0.10 Other functions

Not documented because still in beta version: `parabola`, `Cramer22`, `Cramer33`.

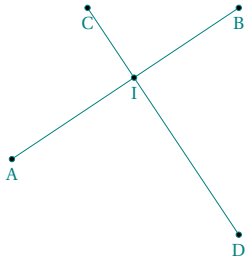
11 Intersections

It's an essential tool. For the moment, the classes concerned are lines, circles and ellipses, with the following combinations: line-line; line-circle; circle-circle and line-ellipse. The argument is a pair of objects, in any order. Results consist of one or two values, either points, boolean **false** or underscore _.

11.1 Line-line

The result is of the form: `point` or `false`.

```
\begin{tkzelements}
   z.A   = point : new (1,-1)
   z.B   = point : new (4,1)
   z.C   = point : new (2,1)
   z.D   = point : new (4,-2)
    z.I  = point : new (0,0)
   L.AB  = line : new (z.A,z.B)
   L.CD  = line : new (z.C,z.D)
   x     = intersection (L.AB,L.CD)
   if x  == false then
   tex.print('error')
   else
   z.I   = x
   end
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawSegments(A,B C,D)
   \tkzDrawPoints(A,B,C,D,I)
   \tkzLabelPoints(A,B,C,D,I)
\end{tikzpicture}
```

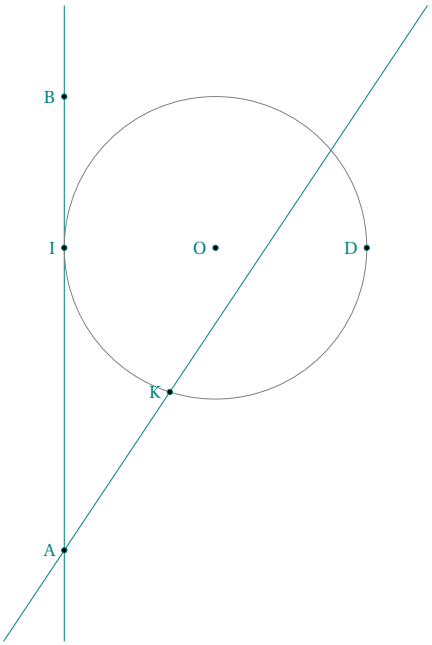


Other examples: 7.2.1, 7.2.2, 12.3, 3.1

11.2 Line-circle

The result is of the form : `point,point` or `false,false`. If the line is tangent to the circle, then the two points are identical. You can ignore one of the points by using the underscore: `_,point` or `point,_`. When the intersection yields two solutions, the order of the points is determined by the argument of `(z.p - z.c)` with `c` center of the circle and `p` point of intersection. The first solution corresponds to the smallest argument (arguments are between 0 and 2π).

```
\begin{tkzelements}
   z.A   = point : new (1,-1)
   z.B   = point : new (1,2)
   L.AB  = line : new (z.A,z.B)
   z.O   = point : new (2,1)
   z.D   = point : new (3,1)
   z.E   = point : new (3,2)
   L.AE  = line : new (z.A,z.E)
   C.OD  = circle : new (z.O,z.D)
   z.I,_ = intersection (L.AB,C.OD)
   _,z.K = intersection (C.OD,L.AE)
\end{tkzelements}

\begin{tikzpicture}
\tkzGetNodes
   \tkzDrawLines(A,B A,E)
   \tkzDrawCircle(O,D)
   \tkzDrawPoints(A,B,O,D,I,K)
   \tkzLabelPoints[left](A,B,O,D,I,K)
\end{tikzpicture}
```

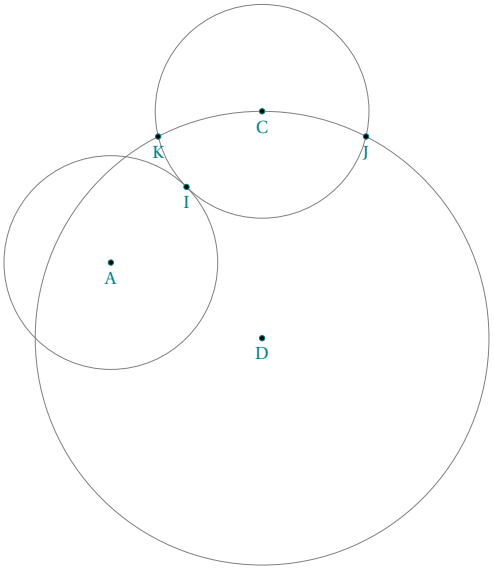


Other examples: 7.2.1

11.3 Circle-circle

The result is of the form : `point,point` or `false,false`. If the circles are tangent, then the two points are identical. You can ignore one of the points by using the underscore: `_ , point` or `point , _`. As for the intersection of a line and a circle, consider the argument of `z.p-z.c` with `c` center of the first circle and `p` point of intersection. The first solution corresponds to the smallest argument (arguments are between 0 and 2π).

```
\begin{tkzelements}
   z.A      = point : new (1,1)
   z.B      = point : new (2,2)
   z.C      = point : new (3,3)
   z.D      = point : new (3,0)
   C.AB     = circle : new (z.A,z.B)
   C.CB     = circle : new (z.C,z.B)
   z.I,_    = intersection (C.AB,C.CB)
   C.DC     = circle : new (z.D,z.C)
   z.J,z.K  = intersection (C.DC,C.CB)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(A,B C,B D,C)
   \tkzDrawPoints(A,I,C,D,J,K)
   \tkzLabelPoints(A,I,C,D,J,K)
\end{tikzpicture}
```

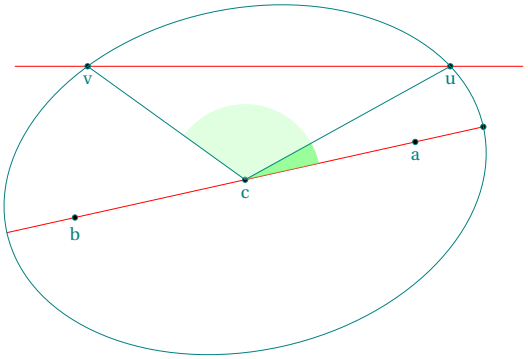


Other examples: 7.2.1, 1.3

11.4 Line-ellipse

The following example is complex, but it shows the possibilities of Lua. The designation of intersection points is a little more complicated than the previous one, as the argument characterizing the major axis must be taken into account. The principle is the same, but this argument must be subtracted. In concrete terms, you need to consider the slopes of the lines formed by the center of the ellipse and the points of intersection, and the slope of the major axis.

```
\begin{tkzelements}
   scale    = .5
   z.a      = point: new (5 , 2)
   z.b      = point: new (-4 , 0)
   z.m      = point: new (2 , 4)
   z.n      = point: new (4 , 4)
   L.ab     = line : new (z.a,z.b)
   L.mn     = line : new (z.m,z.n)
   z.c      = L.ab. mid
   z.e      = L.ab: point (-.2)
   E        = ellipse: foci (z.a,z.b,z.e)
   z.u,z.v  = intersection (E,L.mn)
  -- transfer to tex
   a        = E.Rx
   b        = E.Ry
   ang      = math.deg(E.slope)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines[red](a,b u,v) % p,s p,t
   \tkzDrawPoints(a,b,c,e,u,v) %
   \tkzLabelPoints(a,b,c,u,v)
   \tkzDrawEllipse[teal](c,\tkzUseLua{a},\tkzUseLua{b},\tkzUseLua{ang})
   \tkzDrawSegments(c,u c,v)
   \tkzFillAngles[green!30,opacity=.4](e,c,v)
   \tkzFillAngles[green!80,opacity=.4](e,c,u)
\end{tikzpicture}
```

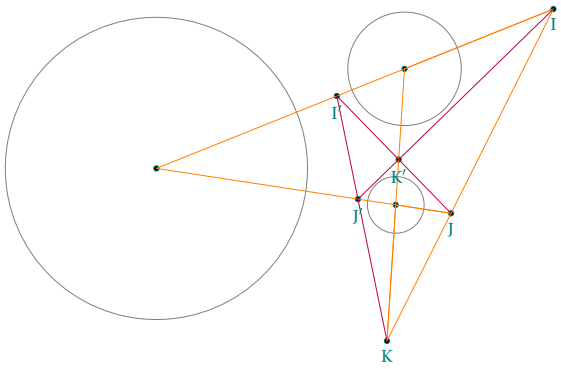


Other examples: 9.2.2, 12.37

12 Examples

12.1 D'Alembert 1

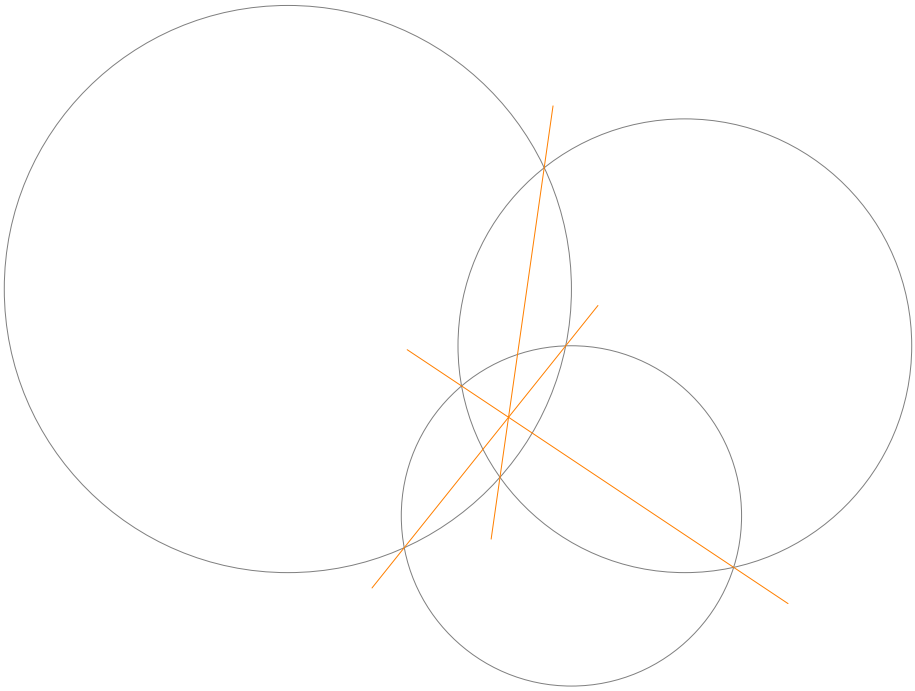
```
\begin{tkzelements}
   z.A = point : new (0,0)
   z.a = point : new (4,0)
   z.B = point : new (7,-1)
   z.b = point : new (5.5,-1)
   z.C = point : new (5,-4)
   z.c = point : new (4.25,-4)
   C.Aa    = circle :    new (z.A,z.a)
   C.Bb    = circle :    new (z.B,z.b)
   C.Cc    = circle :    new (z.C,z.c)
   z.I = C.Aa : external_similitude (C.Bb)
   z.J = C.Aa : external_similitude (C.Cc)
   z.K = C.Cc : external_similitude (C.Bb)
   z.Ip    = C.Aa : internal_similitude (C.Bb)
   z.Jp    = C.Aa : internal_similitude (C.Cc)
   z.Kp    = C.Cc : internal_similitude (C.Bb)
\end{tkzelements}
\begin{tikzpicture}[rotate=-60]
   \tkzGetNodes
   \tkzDrawCircles(A,a B,b C,c)
   \tkzDrawPoints(A,B,C,I,J,K,I',J',K')
   \tkzDrawSegments[new](I,K A,I A,J B,I B,K C,J C,K)
   \tkzDrawSegments[purple](I,J' I',J I',K)
   \tkzLabelPoints(I,J,K,I',J',K')
\end{tikzpicture}
```



12.2 D'Alembert 2

```
\begin{tkzelements}
   scale = .75
   z.A     = point : new (0,0)
   z.a     = point : new (5,0)
   z.B     = point : new (7,-1)
   z.b     = point : new (3,-1)
   z.C     = point : new (5,-4)
   z.c     = point : new (2,-4)
   C.Aa    = circle :    new (z.A,z.a)
   C.Bb    = circle :    new (z.B,z.b)
```

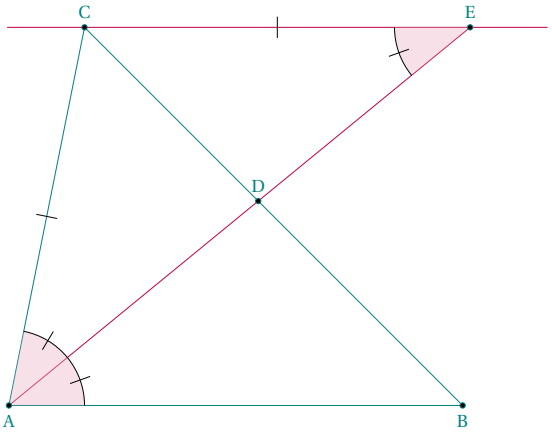
```
   C.Cc    = circle :    new (z.C,z.c)
   z.i,z.j = get_points (C.Aa : radical_axis (C.Bb))
   z.k,z.l = get_points (C.Aa : radical_axis (C.Cc))
   z.m,z.n = get_points (C.Bb : radical_axis (C.Cc))
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(A,a B,b C,c)
   \tkzDrawLines[new](i,j k,l m,n)
\end{tikzpicture}
```



12.3 Alternate

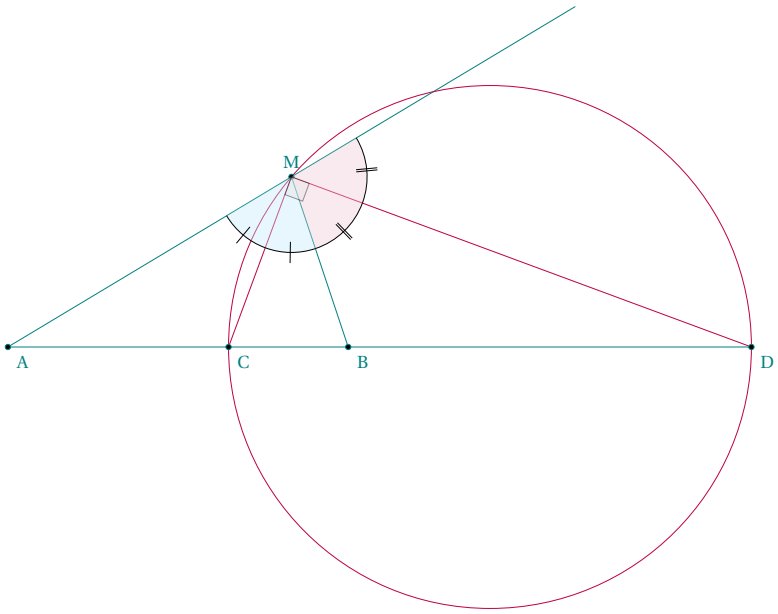
```
\begin{tkzelements}
   z.A  = point: new (0 , 0)
   z.B  = point: new (6 , 0)
   z.C  = point: new (1 , 5)
   T    = triangle: new (z.A,z.B,z.C)
   z.I  = T.incenter
   L.AI = line: new (z.A,z.I)
   z.D  = intersection (L.AI,T.bc)
   L.LLC    = T.ab: ll_from (z.C)
   z.E  = intersection (L.AI,L.LLC)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(A,B,C)
   \tkzDrawLine[purple](C,E)
   \tkzDrawSegment[purple](A,E)
   \tkzFillAngles[purple!30,opacity=.4](B,A,C C,E,D)
   \tkzMarkAngles[mark=|](B,A,D D,A,C C,E,D)
   \tkzDrawPoints(A,...,E)
   \tkzLabelPoints(A,B)
```

```
   \tkzLabelPoints[above](C,D,E)
   \tkzMarkSegments(A,C C,E)
\end{tikzpicture}
```



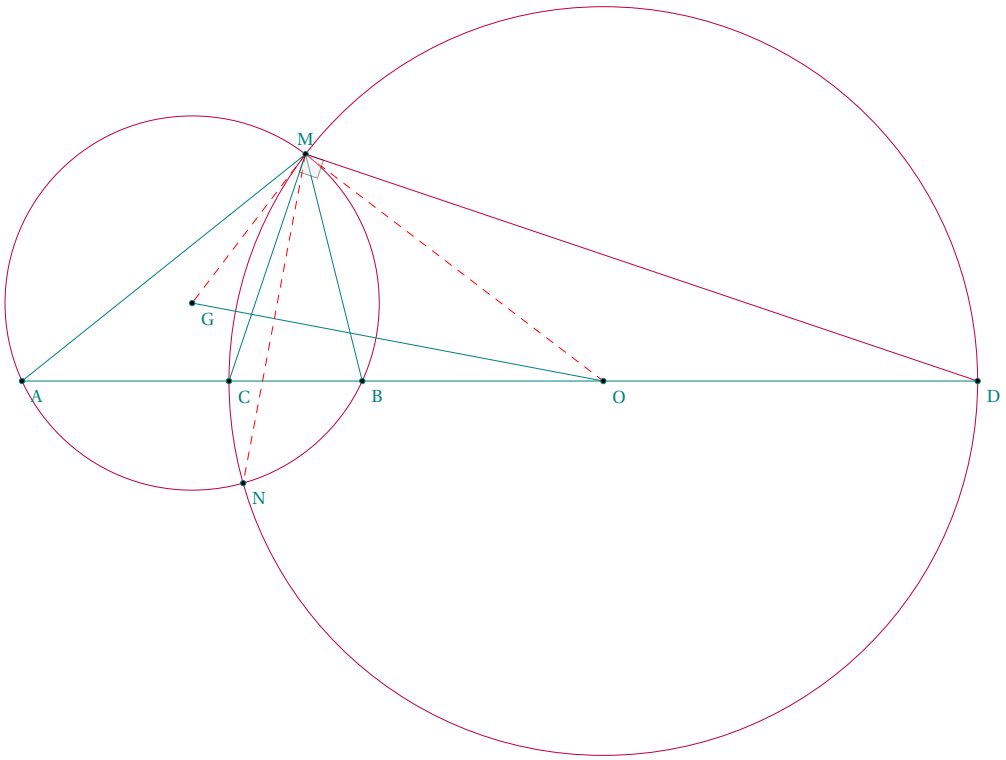
12.4 Apollonius circle

```
\begin{tkzelements}
scale=.75
   z.A      = point: new (0 , 0)
   z.B      = point: new (6 , 0)
   z.M      = point: new (5 , 3)
   T.MAB    = triangle : new (z.M,z.A,z.B)
   L.bis    = T.MAB : bisector ()
   z.C      = L.bis.pb
   L.bisext = T.MAB : bisector_ext ()
   z.D      = intersection (T.MAB.bc, L.bisext)
   L.CD     = line: new (z.C,z.D)
   z.O      = L.CD.mid
   L.AM     = T.MAB.ab
   z.E      = z.M : symmetry (z.A)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawSegment[add=0 and 1](A,M)
   \tkzDrawSegments[purple](M,C M,D)
   \tkzDrawCircle[purple](O,C)
   \tkzDrawSegments(A,B B,M D,B)
   \tkzDrawPoints(A,B,M,C,D)
   \tkzLabelPoints[below right](A,B,C,D)
   \tkzLabelPoints[above](M)
   \tkzFillAngles[opacity=.4,cyan!20](A,M,B)
   \tkzFillAngles[opacity=.4,purple!20](B,M,E)
   \tkzMarkRightAngle[opacity=.4,fill=gray!20](C,M,D)
   \tkzMarkAngles[mark=|](A,M,C C,M,B)
   \tkzMarkAngles[mark=||](B,M,D D,M,E)
\end{tikzpicture}
```



12.5 Apollonius and circle circumscribed

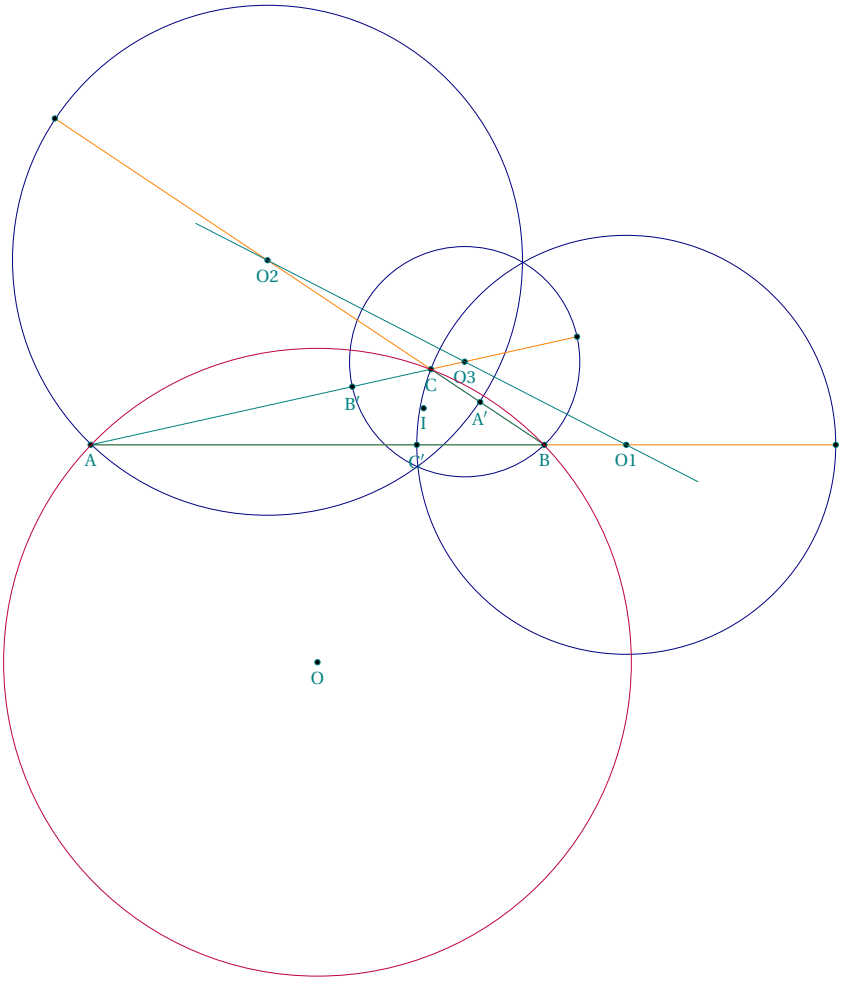
```
\begin{tkzelements}
   scale =.75
   z.A   = point: new (0 , 0)
   z.B   = point: new (6 , 0)
   L.AB  = line: new (z.A,z.B)
   z.M   = point: new (5 , 4)
   T.AMB = triangle: new (z.A,z.M,z.B)
   z.I   = T.AMB.incenter
   L.MI  = line: new (z.M,z.I)
   z.C   = intersection (L.AB , L.MI)
   z.J   = L.MI: ortho_from (z.M)
   L.MJ  = line: new (z.M,z.J)
   z.D   = intersection (L.AB , L.MJ)
   L.CD  = line: new (z.C,z.D)
   z.O   = L.CD.mid
   z.G   = T.AMB.circumcenter
   C.GA  = circle: new (z.G,z.A)
   C.OC  = circle: new (z.O,z.C)
   _,z.N = intersection (C.GA , C.OC)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(A,B,M)
   \tkzDrawCircles[purple](O,C G,A)
   \tkzDrawSegments[purple](M,D)
   \tkzDrawSegments(D,B O,G M,C)
   \tkzDrawSegments[red,dashed](M,N M,O M,G)
   \tkzDrawPoints(A,B,M,C,D,N,O,G)
   \tkzLabelPoints[below right](A,B,C,D,N,O,G)
   \tkzLabelPoints[above](M)
   \tkzMarkRightAngle[opacity=.4,fill=gray!20](C,M,D)
\end{tikzpicture}
```



12.6 Apollonius circles in a triangle

```
\begin{tkzelements}
   z.A   = point: new (0 , 0)
   z.B   = point: new (6 , 0)
   z.C   = point: new (4.5 , 1)
   T.ABC = triangle: new (z.A,z.B,z.C)
   z.I   = T.ABC.incenter
   z.O   = T.ABC.circumcenter
   L.CI  = line: new (z.C,z.I)
   z.Cp  = intersection (T.ABC.ab , L.CI)
   z.x   = L.CI.north_pa
   L.Cx  = line: new (z.C,z.x)
   z.R   = intersection (L.Cx,T.ABC.ab)
   L.CpR = line: new (z.Cp,z.R)
   z.O1  = L.CpR.mid
   L.AI  = line: new (z.A,z.I)
   z.Ap  = intersection (T.ABC.bc , L.AI)
   z.y   = L.AI.north_pa
   L.Ay  = line: new (z.A,z.y)
   z.S   = intersection (L.Ay,T.ABC.bc)
   L.ApS = line: new (z.Ap,z.S)
   z.O2  = L.ApS.mid
   L.BI  = line: new (z.B,z.I)
   z.Bp  = intersection (T.ABC.ac , L.BI)
   z.z   = L.BI.north_pa
   L.Bz  = line: new (z.B,z.z)
   z.T   = intersection (L.Bz,T.ABC.ac)
   L.Bpt = line: new (z.Bp,z.T)
   z.O3  = L.Bpt.mid
```

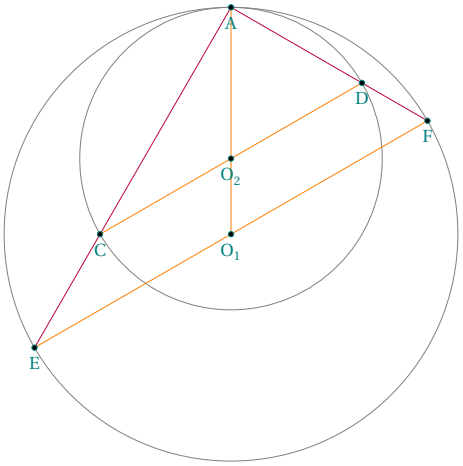
```
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles[blue!50!black](O1,C' O2,A' O3,B')
   \tkzDrawSegments[new](B,S C,T A,R)
   \tkzDrawPolygon(A,B,C)
   \tkzDrawPoints(A,B,C,A',B',C',O,I,R,S,T,O1,O2,O3)
   \tkzLabelPoints(A,B,C,A',B',C',O,I)
   \tkzLabelPoints(O1,O2,O3)
   \tkzDrawCircle[purple](O,A)
   \tkzDrawLine(O1,O2)
\end{tikzpicture}
```



12.7 Archimedes

```
\begin{tkzelements}
   z.O_1    = point:   new  (0, 0)
   z.O_2    = point:   new  (0, 1)
   z.A      = point:   new  (0, 3)
   z.F      = point:   polar (3, math.pi/6)
   L        = line:    new   (z.F,z.O_1)
   C        = circle:  new  (z.O_1,z.A)
   z.E      = intersection (L,C)
```

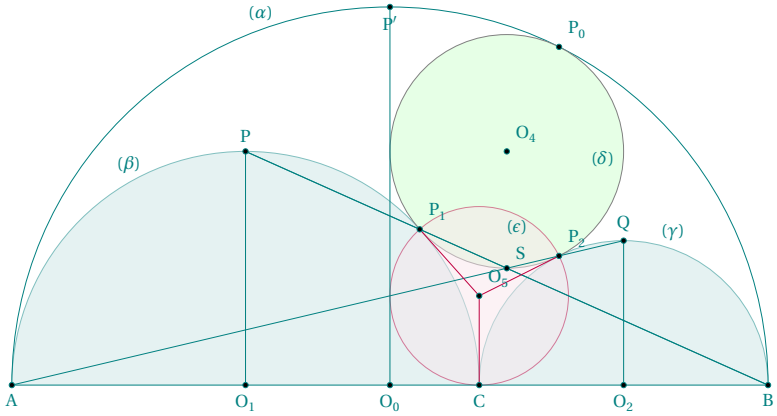
```
   T        = triangle: new (z.F,z.E,z.O_2)
   z.x      = T: parallelogram ()
   L        = line: new    (z.x,z.O_2)
   C        = circle: new  (z.O_2,z.A)
   z.C,z.D  = intersection (L ,C)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(O_1,A O_2,A)
   \tkzDrawSegments[new](O_1,A E,F C,D)
   \tkzDrawSegments[purple](A,E A,F)
   \tkzDrawPoints(A,O_1,O_2,E,F,C,D)
   \tkzLabelPoints(A,O_1,O_2,E,F,C,D)
\end{tikzpicture}
```



12.8 Bankoff circle

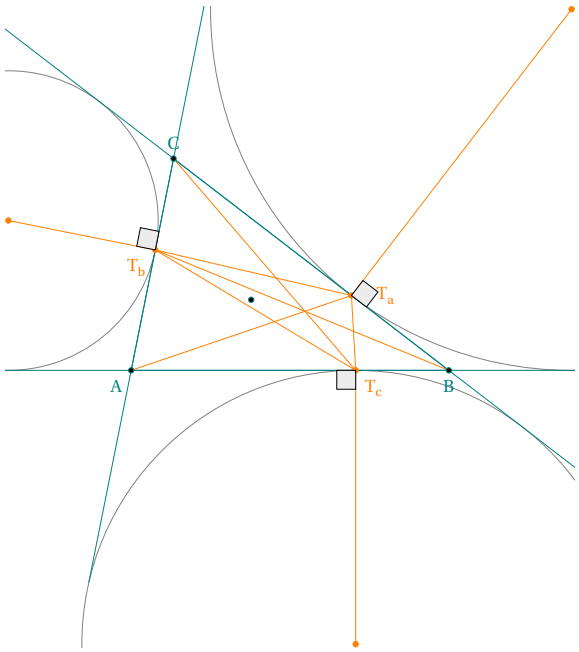
```
\begin{tkzelements}
   z.A      = point: new (0 , 0)
   z.B      = point: new (10 , 0)
   L.AB     = line : new (z.A,z.B)
   z.C      = L.AB: gold_ratio ()
   L.AC     = line : new (z.A,z.C)
   L.CB     = line : new (z.C,z.B)
   z.O_0    = L.AB.mid
   z.O_1    = L.AC.mid
   z.O_2    = L.CB.mid
   C.O0B    = circle : new (z.O_0,z.B)
   C.O1C    = circle : new (z.O_1,z.C)
   C.O2C    = circle : new (z.O_2,z.B)
   z.Pp     = C.O0B : midarc (z.B,z.A)
   z.P      = C.O1C : midarc (z.C,z.A)
   z.Q      = C.O2C : midarc (z.B,z.C)
   L.O1O2   = line : new (z.O_1,z.O_2)
   L.O0O1   = line : new (z.O_0,z.O_1)
   L.O0O2   = line : new (z.O_0,z.O_2)
   z.M_0    = L.O1O2 : harmonic_ext (z.C)
   z.M_1    = L.O0O1 : harmonic_int (z.A)
   z.M_2    = L.O0O2 : harmonic_int (z.B)
   L.BP     = line : new (z.B,z.P)
```

```
   L.AQ     = line : new (z.A,z.Q)
   z.S      = intersection (L.BP,L.AQ)
   L.PpO0   = line : new (z.Pp,z.O_0)
   L.PC     = line : new (z.P,z.C)
   z.Ap     = intersection (L.PpO0,L.PC)
   L.CS     = line : new (z.C,z.S)
   C.M1A    = circle : new (z.M_1,z.A)
   C.M2B    = circle : new (z.M_2,z.B)
   z.P_0    = intersection (L.CS,C.O0B)
   z.P_1    = intersection (C.M2B,C.O1C)
   z.P_2    = intersection (C.M1A,C.O2C)
   T.P0P1P2 = triangle : new (z.P_0,z.P_1,z.P_2)
   z.O_4    = T.P0P1P2.circumcenter
   T.CP1P2  = triangle : new (z.C,z.P_1,z.P_2)
   z.O_5    = T.CP1P2.circumcenter
\end{tkzelements}
\begin{tikzpicture}
\tkzGetNodes
\tkzDrawSemiCircles[teal](O_0,B)
\tkzDrawSemiCircles[teal,fill=teal!20,opacity=.5](O_1,C O_2,B)
\tkzDrawCircle[fill=green!10](O_4,P_0)
\tkzDrawCircle[purple,fill=purple!10,opacity=.5](O_5,C)
\tkzDrawSegments(A,B O_0,P' B,P A,Q)
\tkzDrawSegments(P,B Q,O_2 P,O_1)
\tkzDrawSegments[purple](O_5,P_2 O_5,P_1 O_5,C)
\tkzDrawPoints(A,B,C,P_0,P_2,P_1,O_0,O_1,O_2,O_4,O_5,Q,P,P',S)
\tkzLabelPoints[below](A,B,C,O_0,O_1,O_2,P')
\tkzLabelPoints[above](Q,P)
\tkzLabelPoints[above right](P_0,P_2,P_1,O_5,O_4,S)
\begin{scope}[font=\scriptsize]
  \tkzLabelCircle[above](O_1,C)(120){$(\beta)$}
  \tkzLabelCircle[above](O_2,B)(70){$(\gamma)$}
  \tkzLabelCircle[above](O_0,B)(110){$(\alpha)$}
  \tkzLabelCircle[left](O_4,P_2)(60){$(\delta)$}
  \tkzLabelCircle[left](O_5,C)(140){$(\epsilon)$}
\end{scope}
\end{tikzpicture}
```



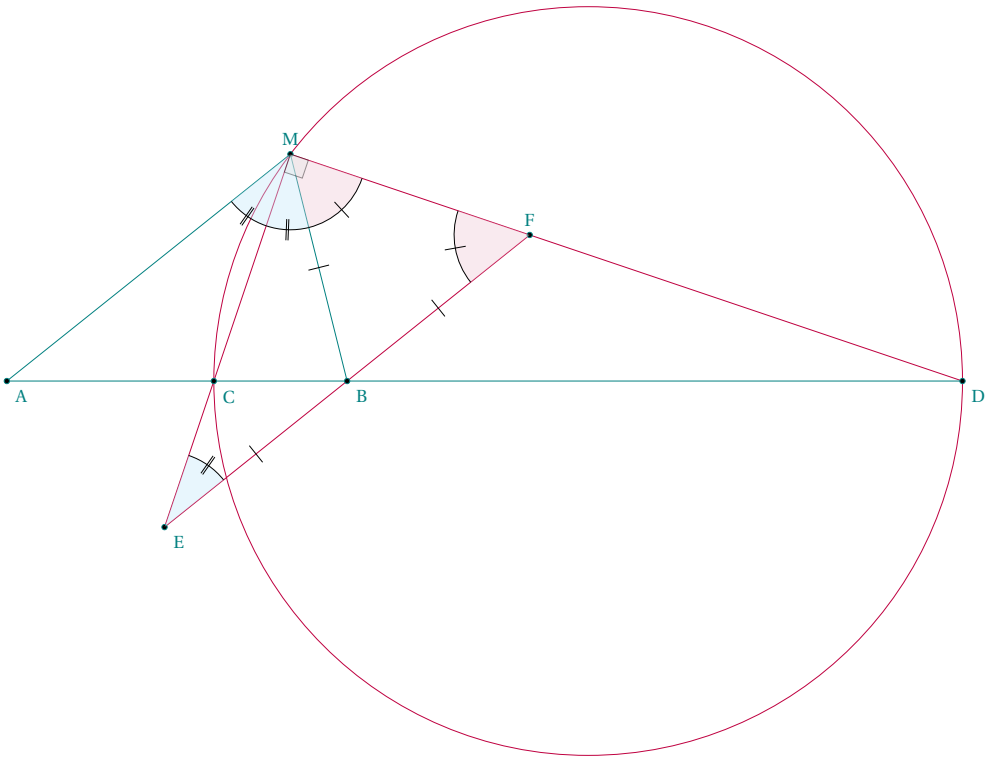
12.9 Excircles

```
\begin{tkzelements}
   scale                = 0.7
   z.A                  = point: new (0,0)
   z.B                  = point: new (6,0)
   z.C                  = point: new (.8,4)
   T                    = triangle: new ( z.A, z.B, z.C)
   z.K                  = T.centroid
   z.J_a,z.J_b,z.J_c    = get_points (T: excentral())
   z.T_a,z.T_b,z.T_c    = get_points (T: extouch())
   la                   = line: new ( z.A, z.T_a)
   lb                   = line: new ( z.B, z.T_b)
   z.G                  = intersection (la,lb)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPoints[new](J_a,J_b,J_c)
   \tkzClipBB
   \tkzDrawCircles[gray](J_a,T_a J_b,T_b J_c,T_c)
   \tkzDrawLines[add=1 and 1](A,B B,C C,A)
   \tkzDrawSegments[new](A,T_a B,T_b C,T_c)
   \tkzDrawSegments[new](J_a,T_a J_b,T_b J_c,T_c)
   \tkzDrawPolygon(A,B,C)
   \tkzDrawPolygon[new](T_a,T_b,T_c)
   \tkzDrawPoints(A,B,C,K)
   \tkzDrawPoints[new](T_a,T_b,T_c)
   \tkzLabelPoints[below left](A)
   \tkzLabelPoints[below](B)
   \tkzLabelPoints[above](C)
   \tkzLabelPoints[new,below left](T_b)
   \tkzLabelPoints[new,below right](T_c)
   \tkzLabelPoints[new,right=6pt](T_a)
   \tkzMarkRightAngles[fill=gray!15](J_a,T_a,B J_b,T_b,C J_c,T_c,A)
\end{tikzpicture}
```



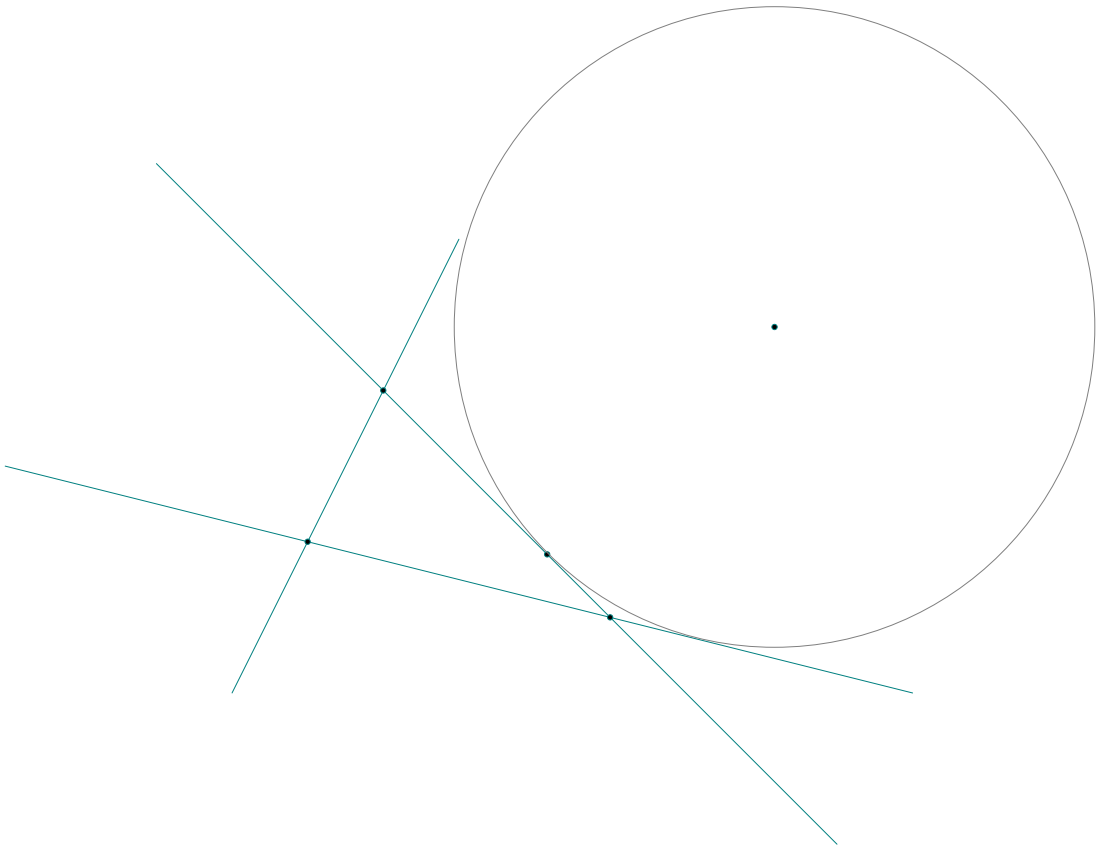
12.10 Harmonic division and bisector

```
\begin{tkzelements}
   scale    =  .75
   z.A      = point: new (0 , 0)
   z.B      = point: new (6 , 0)
   z.M      = point: new (5 , 4)
   T.AMB    = triangle : new (z.A,z.M,z.B)
   L.AB     = T.AMB.ac
   L.bis    = T.AMB : bisector (1)
   z.C      = L.bis.pb
   L.bisext = T.AMB : bisector_ext (1)
   z.D      = intersection (L.bisext,L.AB)
   L.CD     = line: new (z.C,z.D)
   z.O      = L.CD.mid
   L.AM     = line: new (z.A,z.M)
   L.LL     = L.AM : ll_from (z.B)
   L.MC     = line: new (z.M,z.C)
   L.MD     = line: new (z.M,z.D)
   z.E      = intersection (L.LL,L.MC)
   z.F      = intersection (L.LL,L.MD)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(A,B,M)
   \tkzDrawCircle[purple](O,C)
   \tkzDrawSegments[purple](M,E M,D E,F)
   \tkzDrawSegments(D,B)
   \tkzDrawPoints(A,B,M,C,D,E,F)
   \tkzLabelPoints[below right](A,B,C,D,E)
   \tkzLabelPoints[above](M,F)
   \tkzFillAngles[opacity=.4,cyan!20](A,M,B B,E,M)
   \tkzFillAngles[opacity=.4,purple!20](B,M,F M,F,B)
   \tkzMarkRightAngle[opacity=.4,fill=gray!20](C,M,D)
   \tkzMarkAngles[mark=||](A,M,E E,M,B B,E,M)
   \tkzMarkAngles[mark=|](B,M,F M,F,B)
   \tkzMarkSegments(B,E B,M B,F)
\end{tikzpicture}
```



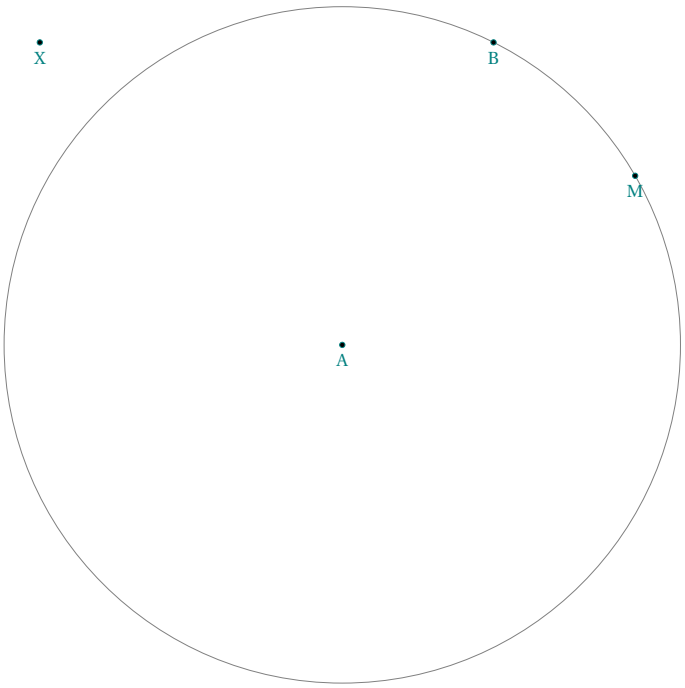
12.11 Excircle

```
\begin{tkzelements}
   z.a     = point:  new (1,2)
   z.b     = point:  new (5,1)
   z.c     = point:  new (2,4)
   T.abc   = triangle: new (z.a,z.b,z.c)
   z.o,z.p = get_points (T.abc: ex_circle ())
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines[add=1 and 1](a,b b,c a,c)
   \tkzDrawPoints(a,b,c,o,p)
   \tkzDrawCircles(o,p)
\end{tikzpicture}
```



12.12 In/Out of a circle or a disk

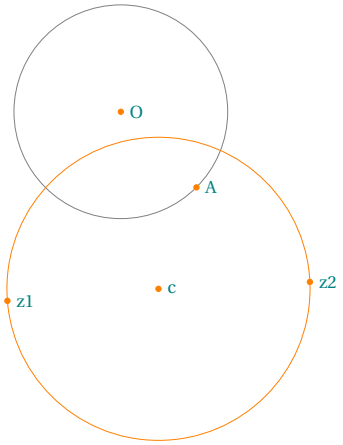
```
\begin{tkzelements}
   scale    = 1.25
   z.A      = point: new (0,0)
   z.B      = point: new (1,2)
   z.X      = point: new (-2,2)
   C        = circle :  new (z.A,z.B)
   if C:in_out(z.X) -- in_out_disk
   then
  tex.print("In")
   else
 tex.print("Out")
   end
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircle(A,B)
   \tkzDrawPoints(A,B,X)
   \tkzLabelPoints(A,B,X)
\end{tikzpicture}
```



Out

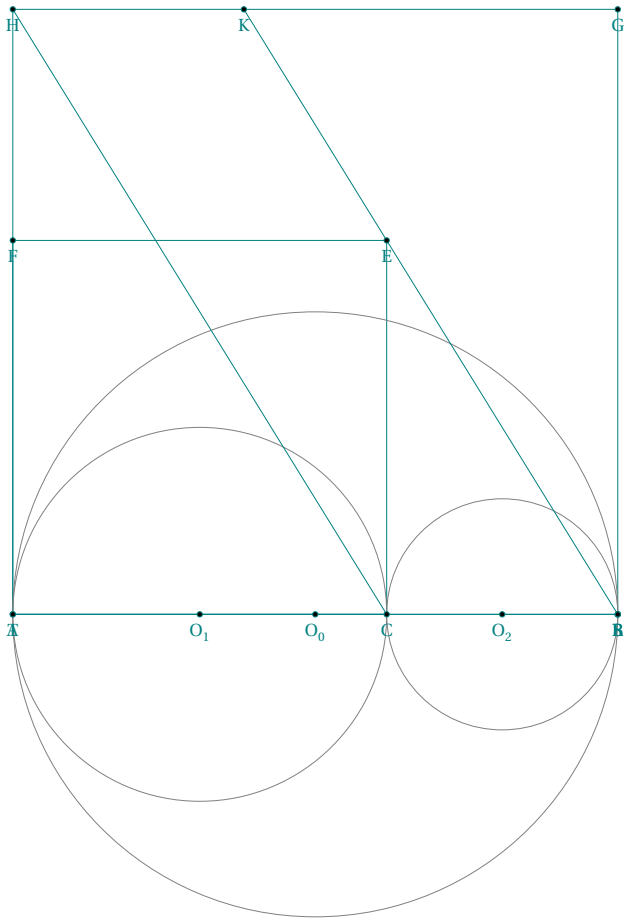
12.13 Orthogonal circle through

```
\begin{tkzelements}
   z.O   = point: new (0,1)
   z.A   = point: new (1,0)
   z.z1  = point: new (-1.5,-1.5)
   z.z2  = point: new (2.5,-1.25)
   C.OA  = circle: new (z.O,z.A)
   C     = C.OA: orthogonal_through (z.z1,z.z2)
   z.c   = C.center
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircle(O,A)
   \tkzDrawCircle[new](c,z1)
   \tkzDrawPoints[new](O,A,z1,z2,c)
   \tkzLabelPoints[right](O,A,z1,z2,c)
\end{tikzpicture}
```



12.14 Devine ratio

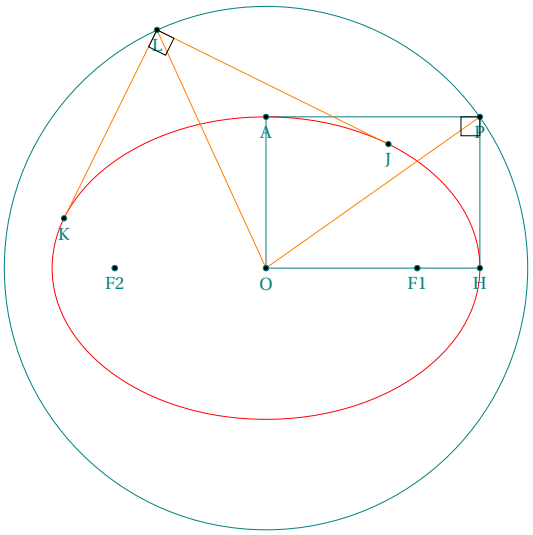
```
\begin{tkzelements}
z.A         = point: new (0 , 0)
z.B         = point: new (8 , 0)
L.AB        = line: new (z.A,z.B)
z.C         = L.AB: gold_ratio ()
L.AC        = line: new (z.A,z.C)
z.O_1       = L.AC.mid
z.G,z.H     = L.AB: square ()
z.E,z.F     = L.AC: square ()
L.CB        = line: new (z.C,z.B)
z.O_2       = L.CB.mid
z.O_0       = L.AB.mid
L.BE        = line: new (z.B,z.E)
L.GH        = line: new (z.G,z.H)
z.K         = intersection (L.BE,L.GH)
C0          = circle: new (z.O_0,z.B)
z.R,_       = intersection (L.BE,C0)
C2          = circle: new (z.O_2,z.B)
z.S,_       = intersection (L.BE,C2)
L.AR        = line:   new (z.A,z.R)
C1          = circle: new (z.O_1,z.C)
_,z.T       = intersection (L.AR,C1)
L.BG        = line: new (z.B,z.G)
z.L         = intersection (L.AR,L.BG)
\end{tkzelements}
\begin{tikzpicture}
\tkzGetNodes
\tkzDrawPolygons(A,C,E,F A,B,G,H)
\tkzDrawCircles(O_1,C O_2,B O_0,B)
\tkzDrawSegments(H,C B,K A,L)
\tkzDrawPoints(A,B,C,K,E,F,G,H,O_0,O_1,O_2,R,S,T,L)
\tkzLabelPoints(A,B,C,K,E,F,G,H,O_0,O_1,O_2,R,S,T,L)
\end{tikzpicture}
```



12.15 Director circle

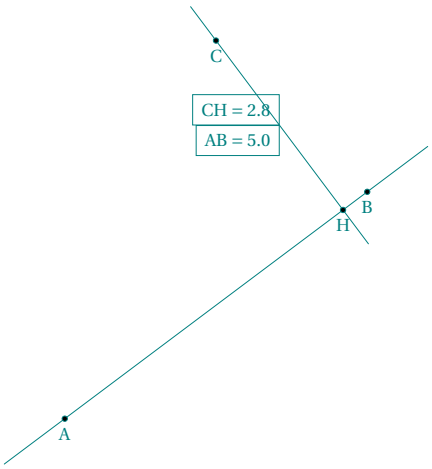
```
\begin{tkzelements}
scale    = .5
z.O      = point: new (0 , 0)
z.F1     = point: new (4 , 0)
z.F2     = point: new (-4 , 0)
z.H      = point: new (4*math.sqrt(2) , 0)
E        = ellipse: foci (z.F2,z.F1,z.H)
a,b      = E.Rx, E.Ry
z.A      = E.covertex
T        = triangle: new (z.H,z.O,z.A)
z.P      = T: parallelogram ()
C        = circle: new (z.O,z.P)
z.L      = C: point (2)
L.J,L.K  = E: tangent_from (z.L)
z.J      = L.J.pb
z.K      = L.K.pb
\end{tkzelements}
 \begin{tikzpicture}
 \tkzGetNodes
 \tkzDrawPoints(F1,F2,O)
\tkzDrawCircles[teal](O,P)
\tkzDrawPolygon(H,O,A,P)
\tkzDrawEllipse[red](O,\tkzUseLua{a},\tkzUseLua{b},0)
```

```
\tkzDrawSegments[orange](O,P O,L L,J L,K)
\tkzDrawPoints(F1,F2,O,H,A,P,L,J,K)
\tkzLabelPoints(F1,F2,O,H,A,P,L,J,K)
\tkzMarkRightAngles(A,P,H J,L,K)
\end{tikzpicture}
```



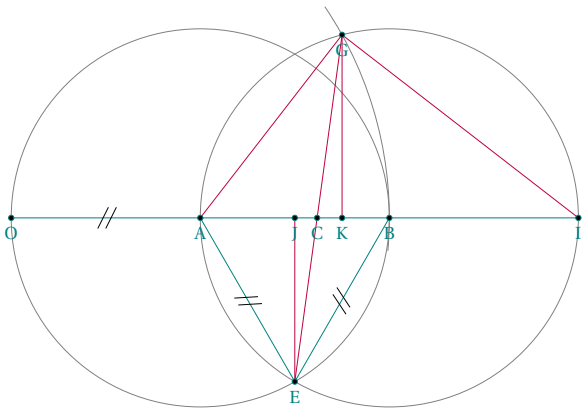
12.16 Distance

```
\begin{tkzelements}
   z.A      = point : new (0 , 0)
   z.B      = point : new (4 , 3)
   z.C      = point : new (2 , 5)
   L.AB     = line : new (z.A,z.B)
   d        = L.AB : distance (z.C)
   l        = L.AB.length
   z.H      = L.AB : projection (z.C)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines(A,B C,H)
   \tkzDrawPoints(A,B,C,H)
   \tkzLabelPoints(A,B,C,H)
   \tkzLabelSegment[above left,draw](C,H){$CH = \tkzUseLua{d}$}
   \tkzLabelSegment[below left,draw](C,H){$AB =\tkzUseLua{l}$}
\end{tikzpicture}
```



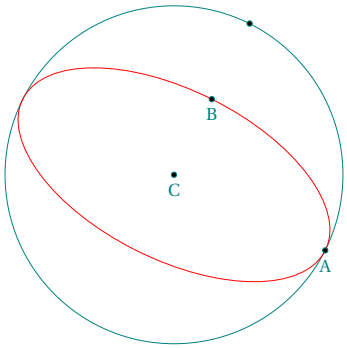
12.17 Gold division

```
\begin{tkzelements}
   z.A      = point:  new (0,0)
   z.B      = point:  new (2.5,0)
   L.AB     = line:   new (z.A,z.B)
   C.AB     = circle: new (z.A,z.B)
   C.BA     = circle: new (z.B,z.A)
   z.J      = L.AB: midpoint ()
   L.JB     = line:new (z.J,z.B)
   z.F,z.E  = intersection (C.AB , C.BA)
   z.I,_    = intersection (L.AB , C.BA)
   z.K      = L.JB : midpoint ()
   z.k      = L.JB: ortho_from (z.K)
   L.Kk     = line:new (z.K,z.k)
   _,z.G    = intersection (L.Kk,C.BA)
   L.EG     = line:new (z.E,z.G)
   z.C      = intersection  (L.EG,L.AB)
   z.O      = C.AB: antipode (z.B)
\end{tkzelements}
\begin{tikzpicture}
\tkzGetNodes
   \tkzDrawArc[delta=5](O,B)(G)
   \tkzDrawCircles(A,B B,A)
   \tkzDrawSegments(A,E B,E O,I)
   \tkzDrawSegments[purple](J,E A,G G,I K,G E,G)
   \tkzMarkSegments[mark=s||](A,E B,E O,A)
   \tkzDrawPoints(A,B,C,E,I,J,G,O,K)
   \tkzLabelPoints(A,B,C,E,I,J,G,O,K)
\end{tikzpicture}
```



12.18 Ellipse

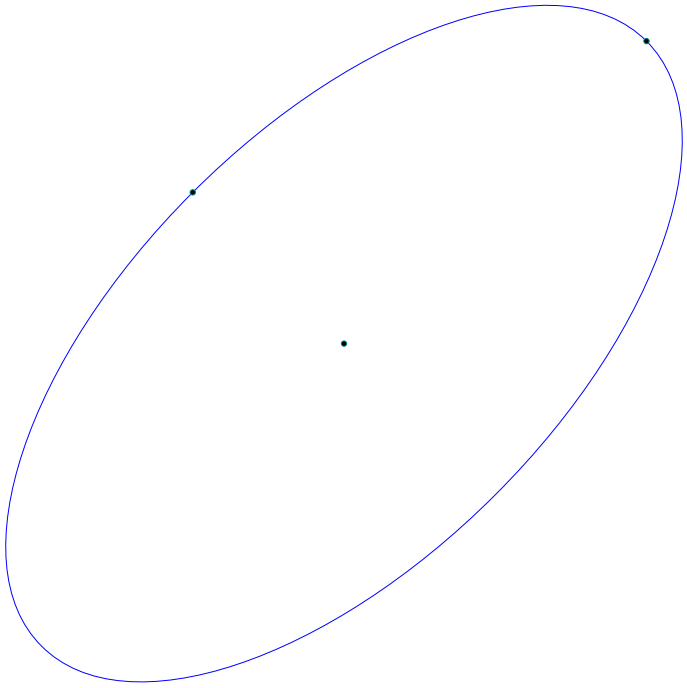
```
\begin{tkzelements}
   z.C       = point: new (3 , 2)
   z.A       = point: new (5 , 1)
   L.CA      = line : new (z.C,z.A)
   z.b       = L.CA.north_pa
   L         = line : new (z.C,z.b)
   z.B       = L : point (0.5)
   E         = ellipse: new (z.C,z.A,z.B)
   a         = E.Rx
   b         = E.Ry
   slope     = math.deg(E.slope)

\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles[teal](C,A)
   \tkzDrawEllipse[red](C,\tkzUseLua{a},\tkzUseLua{b},\tkzUseLua{slope})
   \tkzDrawPoints(C,A,B,b)
   \tkzLabelPoints(C,A,B)
\end{tikzpicture}
```



12.19 Ellipse with radii

```
\begin{tkzelements}
z.C      = point: new (0 , 4)
z.B      = point: new (4 , 0)
```

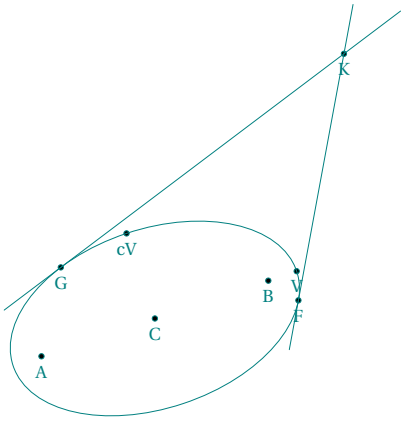
```
z.D      = point: new (2 , 6)
b        = math.sqrt(8)
a        = math.sqrt(32)
ang      = math.deg(math.pi/4)
E        = ellipse: radii (z.C,a,b,math.pi/4)
z.V      = E : point (0)
z.CoV    = E : point (math.pi/2)
\end{tkzelements}
\begin{tikzpicture}
\tkzGetNodes
\tkzDrawEllipse[blue](C,\tkzUseLua{a},\tkzUseLua{b},\tkzUseLua{ang})
\tkzDrawPoints(C,V,CoV)
\end{tikzpicture}
```



12.20 Ellipse_with_foci

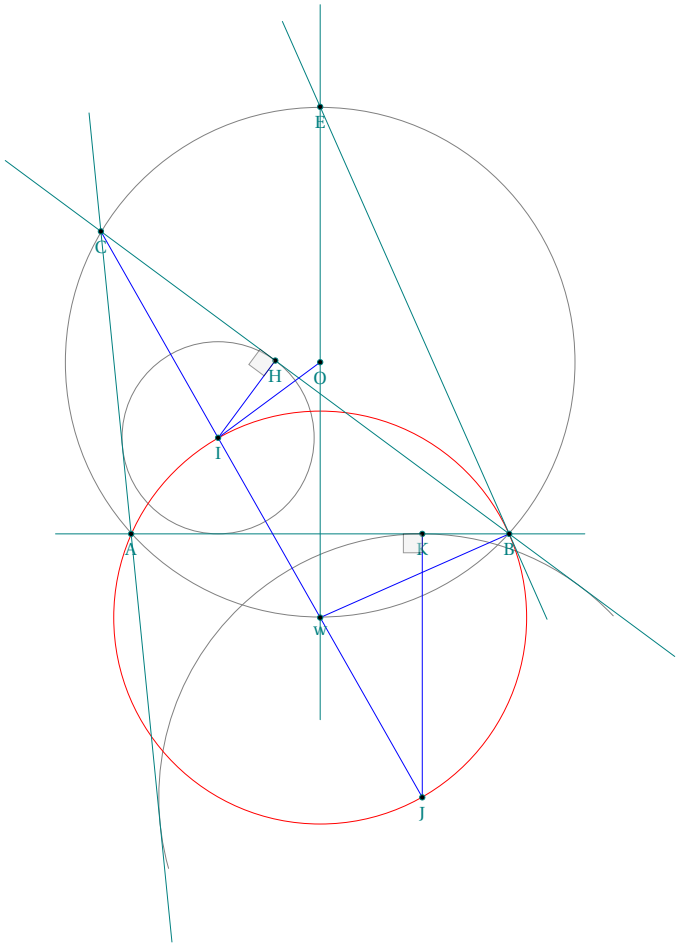
```
\begin{tkzelements}
   local e
   e           = .8
   z.A         = point: new (2 , 3)
   z.B         = point: new (5 , 4)
   z.K         = point: new (6, 7)
   L.AB        = line: new (z.A,z.B)
   z.C         = L.AB.mid
   c           = point.abs(z.B-z.C)
   a           = c/e
   b           = math.sqrt (a^2-c^2)
   z.V         = z.C + a*(z.B-z.C)/point.abs(z.B-z.C)
   E           = ellipse: foci (z.A,z.B,z.V)
   z.cV        = E.covertex
   ang         = math.deg(E.slope)
   L.ta,L.tb   = E: tangent_from (z.K)
   z.F         = L.ta.pb
   z.G         = L.tb.pb
```

```
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPoints(A,B,C,K,F,G,V,cV)
   \tkzLabelPoints(A,B,C,K,F,G,V,cV)
   \tkzDrawEllipse[teal](C,\tkzUseLua{a},\tkzUseLua{b},\tkzUseLua{ang})
   \tkzDrawLines(K,F K,G)
\end{tikzpicture}
```



12.21 Euler relation

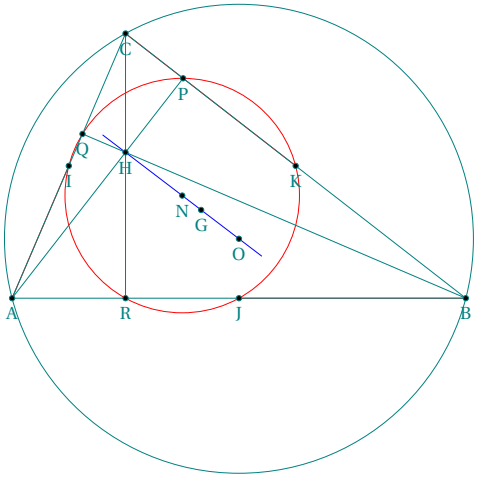
```
\begin{tkzelements}
   scale          =  .75
   z.A            = point: new (0 , 0)
   z.B            = point: new (5 , 0)
   z.C            = point: new (-.4 , 4)
   T.ABC          = triangle: new (z.A,z.B,z.C)
   z.J,z.K        = get_points(T.ABC: ex_circle (2))
   z.X ,z.Y,z.K   = T.ABC : projection (z.J)
   z.I,z.H        = get_points(T.ABC : in_circle())
   z.O            = T.ABC.circumcenter
   C.OA           = circle : new (z.O,z.A)
   T.IBA          = triangle: new (z.I,z.B,z.A)
   z.w            = T.IBA.circumcenter
   L.Ow           = line : new (z.O,z.w)
   _,z.E          = intersection (L.Ow, C.OA)
\end{tkzelements}
   \begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawArc(J,X)(Y)
   \tkzDrawCircles(I,H O,A)
   \tkzDrawCircle[red](w,I)
   \tkzDrawLines(Y,C A,B X,C E,w E,B)
   \tkzDrawSegments[blue](J,C J,K I,H I,O w,B)
   \tkzDrawPoints(A,B,C,I,J,E,w,H,K,O)
   \tkzLabelPoints(A,B,C,J,I,w,H,K,E,O)
   \tkzMarkRightAngles[fill=gray!20,opacity=.4](C,H,I A,K,J)
   \end{tikzpicture}
```



12.22 Euler line

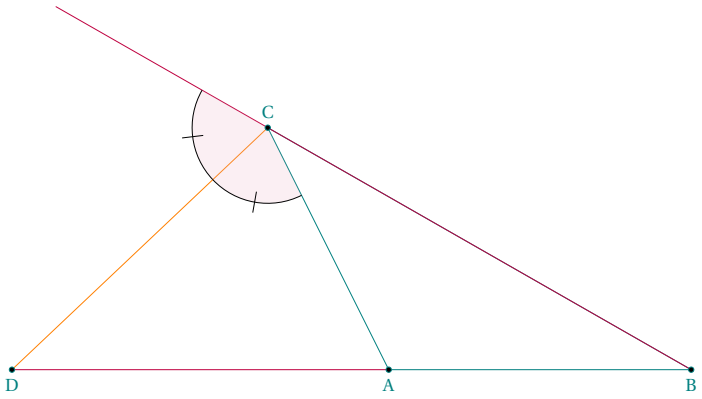
```
\begin{tkzelements}
   z.A           = point: new (0 , 0)
   z.B           = point: new (6 , 0)
   z.C           = point: new (1.5 , 3.5)
   T.ABC         = triangle: new (z.A,z.B,z.C)
   z.O           = T.ABC.circumcenter
   z.G           = T.ABC.centroid
   z.N           = T.ABC. eulercenter
   z.H           = T.ABC. orthocenter
   z.P,z.Q,z.R   = get_points (T.ABC: orthic())
   z.K,z.I,z.J   = get_points (T.ABC: medial ())
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines[blue](O,H)
   \tkzDrawCircle[red](N,I)
   \tkzDrawCircles[teal](O,A)
   \tkzDrawSegments(A,P B,Q C,R)
   \tkzDrawSegments[red](A,I B,J C,K)
   \tkzDrawPolygons(A,B,C)
   \tkzDrawPoints(A,B,C,N,I,J,K,O,P,Q,R,H,G)
   \tkzLabelPoints(A,B,C,I,J,K,P,Q,R,H)
```

```
   \tkzLabelPoints[below](N,O,G)
\end{tikzpicture}
```



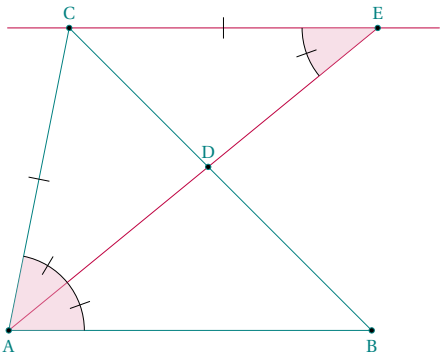
12.23 External angle

```
\begin{tkzelements}
   scale =.8
   z.A    = point: new (0 , 0)
   z.B    = point: new (5 , 0)
   z.C    = point: new (-2 , 4)
   T.ABC  = triangle: new (z.A,z.B,z.C)
   T.ext  = T.ABC: excentral ()
   z.O    = T.ABC.circumcenter
   z.D    = intersection (T.ext.ab,T.ABC.ab)
   z.E    = z.C: symmetry (z.B)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(A,B,C)
   \tkzDrawLine[purple,add=0 and .5](B,C)
   \tkzDrawSegment[purple](A,D)
   \tkzDrawSegment[orange](C,D)
   \tkzFillAngles[purple!30,opacity=.2](D,C,A E,C,D)
   \tkzMarkAngles[mark=|](D,C,A E,C,D)
   \tkzDrawPoints(A,...,D)
   \tkzLabelPoints[above](C)
   \tkzLabelPoints(A,B,D)
\end{tikzpicture}
```



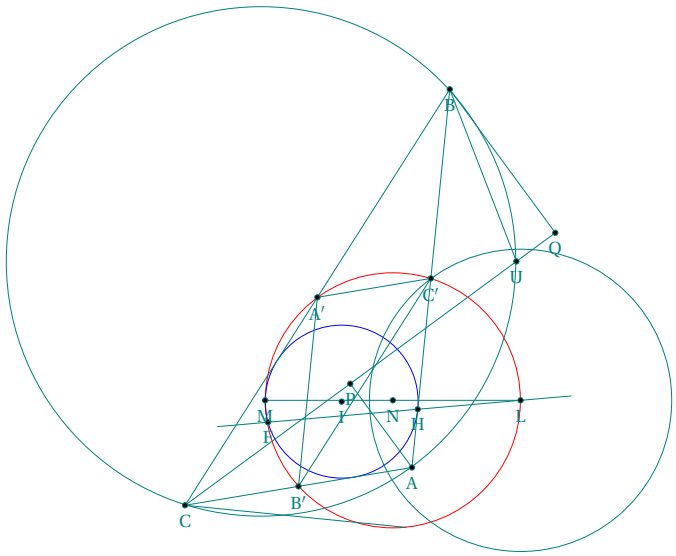
12.24 Internal angle

```
\begin{tkzelements}
   scale =  .8
   z.A   = point: new (0 , 0)
   z.B   = point: new (6 , 0)
   z.C   = point: new (1 , 5)
   T     = triangle: new (z.A,z.B,z.C)
   z.I   = T.incenter
   L.AI  = line: new (z.A,z.I)
   z.D   = intersection (L.AI, T.bc)
   L.LL  = T.ab: ll_from (z.C)
   L.AD  = line: new (z.A,z.D)
   z.E   = intersection (L.LL,L.AD)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(A,B,C)
   \tkzDrawLine[purple](C,E)
   \tkzDrawSegment[purple](A,E)
   \tkzFillAngles[purple!30,opacity=.4](B,A,C C,E,D)
   \tkzMarkAngles[mark=|](B,A,D D,A,C C,E,D)
   \tkzDrawPoints(A,...,E)
   \tkzLabelPoints(A,B)
   \tkzLabelPoints[above](C,D,E)
   \tkzMarkSegments(A,C C,E)
\end{tikzpicture}
```



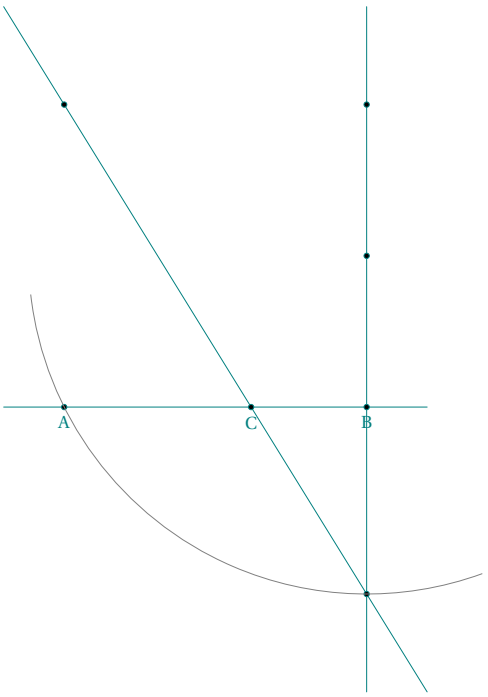
12.25 Feuerbach theorem

```
\begin{tkzelements}
   scale     =  1.5
   z.A       = point: new (0 , 0)
   z.B       = point: new (5 , -.5)
   z.C       = point: new (-.5 , 3)
   T.ABC     = triangle: new (z.A,z.B,z.C)
   z.O       = T.ABC.circumcenter
   z.N       = T.ABC.eulercenter
   z.I,z.K   = get_points(T.ABC: in_circle())
   z.H       = T.ABC.ab : projection (z.I)
   z.Ap,
   z.Bp,
   z.Cp      = get_points (T.ABC : medial ())
   C.IH      = circle:new (z.I,z.H)
   C.NAp     = circle:new (z.N,z.Ap)
   C.OA      = circle:new (z.O,z.A)
   z.U       = C.OA.south
   z.L       = C.NAp.south
   z.M       = C.NAp.north
   z.X       = T.ABC.ab: projection (z.C)
   L.CU      = line: new (z.C,z.U)
   L.ML      = line: new (z.M,z.L)
   z.P       = L.CU: projection (z.A)
   z.Q       = L.CU: projection (z.B)
   L.LH      = line: new (z.L,z.H)
   z.F       = intersection (L.LH,C.IH) -- feuerbach
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLine(L,F)
   \tkzDrawCircle[red](N,A')
   \tkzDrawCircle[blue](I,H)
   \tkzDrawCircles[teal](O,A L,C')
   \tkzDrawSegments(M,L B,U Q,C C,X A,P B,Q)
   \tkzDrawPolygons(A,B,C A',B',C')
   \tkzDrawPoints(A,B,C,N,H,A',B',C',U,L,M,P,Q,F,I)
   \tkzLabelPoints(A,B,C,N,H,A',B',C',U,L,M,P,Q,F,I)
\end{tikzpicture}
```



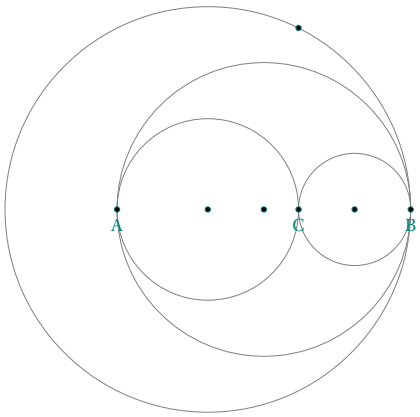
12.26 Gold ratio with segment

```
\begin{tkzelements}
   z.A      = point: new (0 , 0)
   z.B      = point: new (8 , 0)
   L.AB     = line: new (z.A,z.B)
   z.X,z.Y  = L.AB: square ()
   L.BX     = line: new (z.B,z.X)
   z.M      = L.BX.mid
   C.MA     = circle: new (z.M,z.A)
   _,z.K    = intersection (L.BX,C.MA)
   L.AK     = line: new (z.Y,z.K)
   z.C      = intersection (L.AK,L.AB)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines(A,B X,K)
   \tkzDrawLine[teal](Y,K)
   \tkzDrawPoints(A,B,C,X,Y,M,K)
   \tkzDrawArc[delta=20](M,A)(K)
   \tkzLabelPoints(A,B,C)
\end{tikzpicture}
```



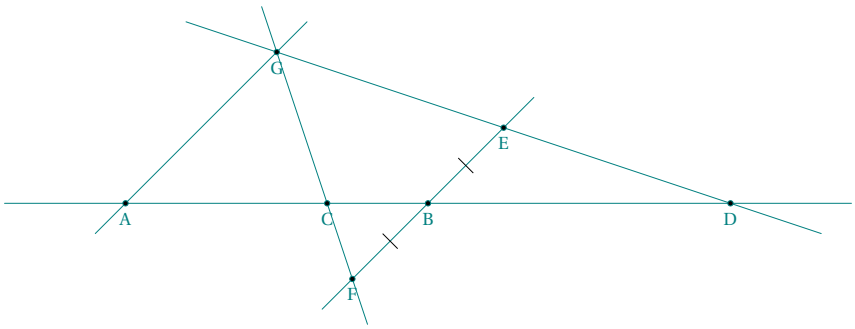
12.27 Gold Arbelos

```
\begin{tkzelements}
   scale    = .6
   z.A      = point: new (0 , 0)
   z.C      = point: new (6 , 0)
   L.AC     = line: new (z.A,z.C)
   z.x,z.y  = L.AC: square ()
   z.O_1    = L.AC . mid
   C        = circle: new (z.O_1,z.x)
   z.B      = intersection (L.AC,C)
   L.CB     = line: new (z.C,z.B)
   z.O_2    = L.CB.mid
   L.AB     = line: new (z.A,z.B)
   z.O_0    = L.AB.mid
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(O_1,C O_2,B O_0,B)
   \tkzDrawPoints(A,C,B,O_1,O_2,O_0)
   \tkzLabelPoints(A,C,B)
\end{tikzpicture}
```



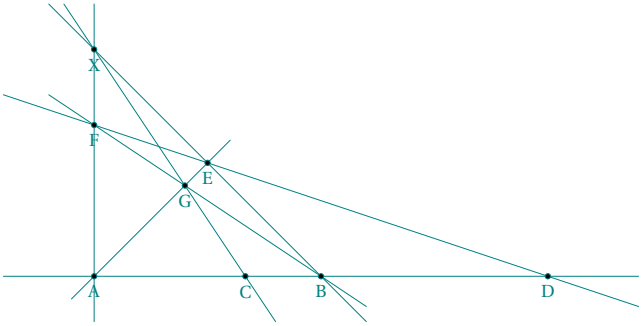
12.28 Harmonic division v1

```
\begin{tkzelements}
   scale=.75
   z.A  = point: new (0 , 0)
   z.B  = point: new (4 , 0)
   z.D  = point: new (12,0)
   L.AB = line : new (z.A,z.B)
   z.X  = L.AB.north_pa
   L.XB = line : new (z.X,z.B)
   z.E  = L.XB.mid
   L.DE = line : new (z.D,z.E)
   L.XA = line : new (z.X,z.A)
   z.F  = intersection (L.DE,L.XA)
   L.AE = line : new (z.A,z.E)
   L.BF = line : new (z.B,z.F)
   z.G  = intersection (L.AE,L.BF)
   L.XG = line : new (z.X,z.G)
z.C  = intersection (L.XG,L.AB)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDefPoints{0/0/A,4/0/B}
   \tkzDefPoints{2/2/G}
   \tkzDefLine[parallel=through B,K=.5](A,G) \tkzGetPoint{E}
   \tkzInterLL(G,E)(A,B)   \tkzGetPoint{D}
   \tkzDefPointBy[symmetry= center B](E) \tkzGetPoint{F}
   \tkzInterLL(G,F)(A,B)   \tkzGetPoint{C}
   \tkzDrawLines(A,D A,G F,E G,F G,D)
   \tkzDrawPoints(A,B,G,E,F,C,D)
   \tkzLabelPoints(A,B,G,E,F,C,D)
   \tkzMarkSegments(F,B B,E)
\end{tikzpicture}
```



12.29 Harmonic division v2

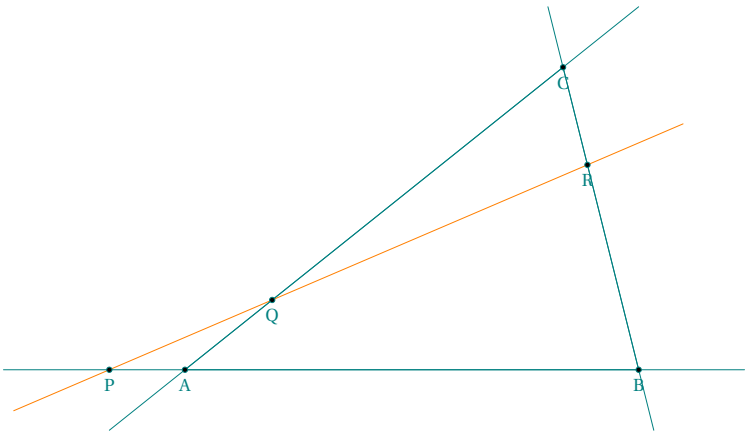
```
\begin{tkzelements}
   scale    = .5
   z.A      = point: new (0 , 0)
   z.B      = point: new (6 , 0)
   z.D      = point: new (12 , 0)
   L.AB     = line: new (z.A,z.B)
   z.X      = L.AB.north_pa
   L.XB     = line: new (z.X,z.B)
   z.E      = L.XB.mid
   L.ED     = line: new (z.E,z.D)
   L.AX     = line: new (z.A,z.X)
   L.AE     = line: new (z.A,z.E)
   z.F      = intersection (L.ED,L.AX)
   L.BF     = line: new (z.B,z.F)
   z.G      = intersection (L.AE,L.BF)
   L.GX     = line: new (z.G,z.X)
   z.C      = intersection (L.GX,L.AB)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines(A,D A,E B,F D,F X,A X,B X,C)
   \tkzDrawPoints(A,...,G,X)
   \tkzLabelPoints(A,...,G,X)
\end{tikzpicture}
```



12.30 Menelaus

```
\begin{tkzelements}
   z.A  = point: new (0 , 0)
   z.B  = point: new (6 , 0)
```

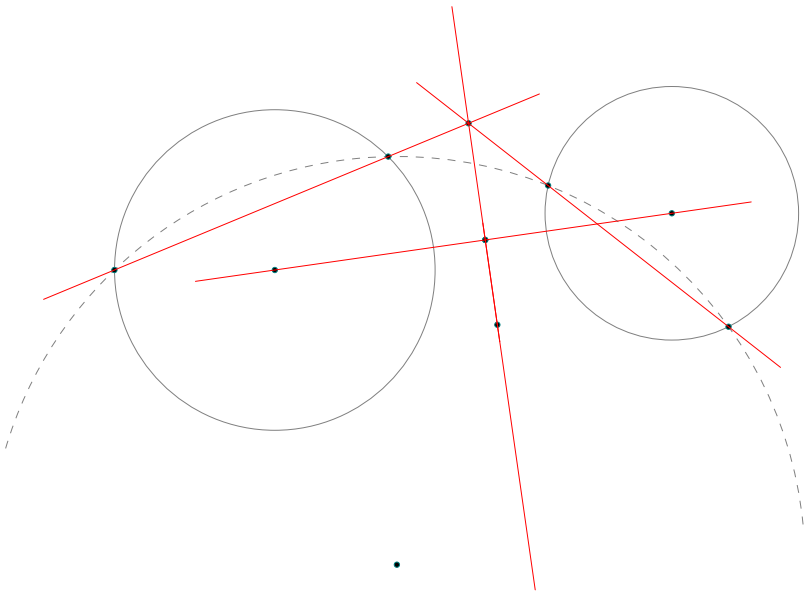
```
   z.C  = point: new (5 , 4)
   z.P  = point: new (-1 , 0)
   z.X  = point: new (6 , 3)
   L.AC = line: new (z.A,z.C)
   L.PX = line: new (z.P,z.X)
   L.BC = line: new (z.B,z.C)
   z.Q  = intersection (L.AC,L.PX)
   z.R  = intersection (L.BC,L.PX)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(A,B,C)
   \tkzDrawLine[new](P,R)
   \tkzDrawLines(P,B A,C B,C)
   \tkzDrawPoints(P,Q,R,A,B,C)
   \tkzLabelPoints(A,B,C,P,Q,R)
\end{tikzpicture}
```



12.31 Radical axis v1

```
\begin{tkzelements}
   scale   = .75
   z.X = point : new (0,0)
   z.B = point : new (2,2)
   z.Y = point : new (7,1)
   z.Ap    = point : new (8,-1)
   L.XY    = line :    new (z.X,z.Y)
   C.XB    = circle : new (z.X,z.B)
   C.YAp   = circle : new (z.Y,z.Ap)
   z.E,z.F = get_points (C.XB : radical_axis (C.YAp))
   z.A = C.XB : point (-math.pi)
   T.ABAp  = triangle: new (z.A,z.B,z.Ap)
   z.O = T.ABAp.circumcenter
   C.OAp   = circle : new (z.O,z.Ap)
   _,z.Bp  = intersection (C.OAp,C.YAp)
   L.AB    = line : new (z.A,z.B)
   L.ApBp  = line : new (z.Ap,z.Bp)
   z.M = intersection (L.AB,L.ApBp)
   z.H = L.XY : projection (z.M)
\end{tkzelements}
\begin{tikzpicture}
```

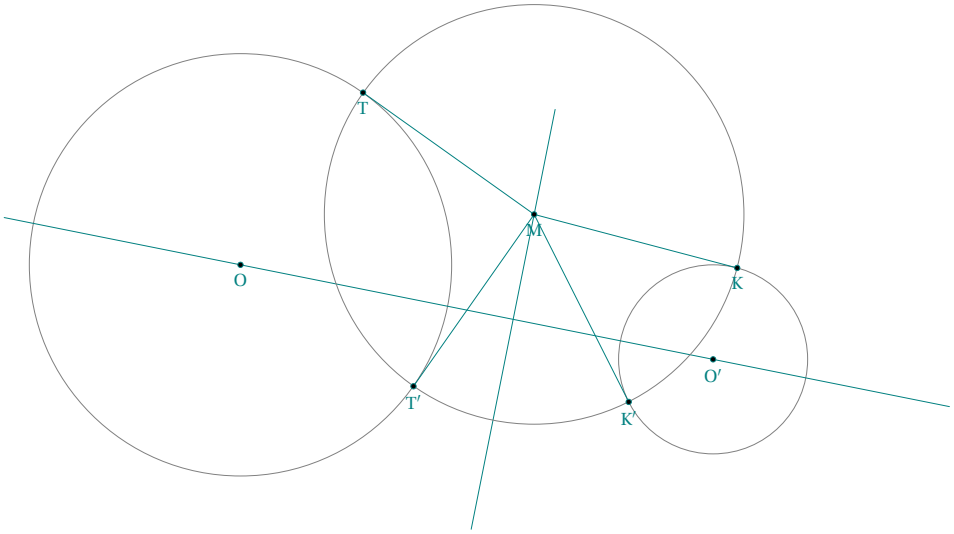
```
   \tkzGetNodes
   \tkzDrawCircles(X,B Y,A')
   \tkzDrawArc[dashed,delta=30](O,A')(A)
   \tkzDrawPoints(A,B,A',B',M,H,X,Y,O,E,F)
   \tkzDrawLines[red](A,M A',M X,Y E,F)
   \tkzDrawLines[red,add=1 and 3](M,H)
\end{tikzpicture}
```



12.32 Radical axis v2

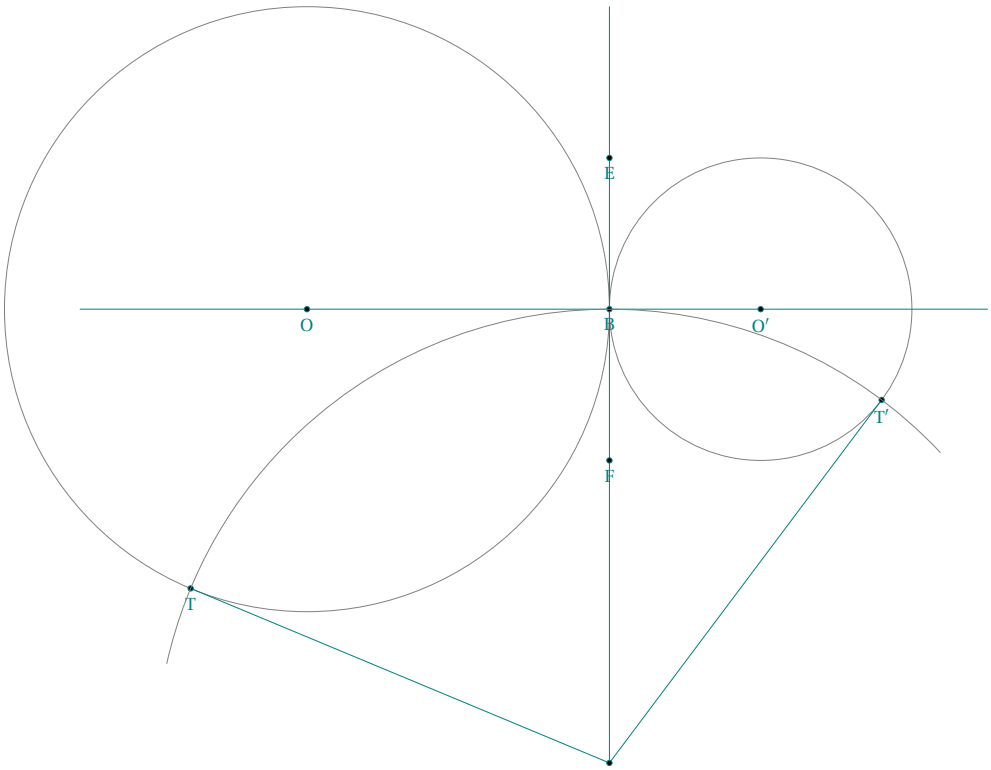
```
\begin{tkzelements}
   z.O          = point : new (-1,0)
   z.Op         = point : new (4,-1)
   z.B          = point : new (0,2)
   z.D          = point : new (4,0)
   C.OB         = circle :    new (z.O,z.B)
   C.OpD        = circle :    new (z.Op,z.D)
   L.EF         = C.OB : radical_axis (C.OpD)
   z.E,z.F      = get_points (L.EF)
   z.M          = L.EF : point (2)
   L.MT,L.MTp   = C.OB : tangent_from (z.M)
   _,z.T        = get_points (L.MT)
   _,z.Tp       = get_points (L.MTp)
   L.MK,L.MKp   = C.OpD : tangent_from (z.M)
   _,z.K        = get_points (L.MK)
   _,z.Kp       = get_points (L.MKp)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(O,B O',D)
   \tkzDrawLine[add=1 and 2](E,F)
   \tkzDrawLine[add=.5 and .5](O,O')
   \tkzDrawSegments(M,T M,T' M,K M,K')
   \tkzDrawCircle(M,T)
   \tkzDrawPoints(O,O',T,M,T',K,K')
   \tkzLabelPoints(O,O',T,T',K,K',M)
```

```
\end{tikzpicture}
```



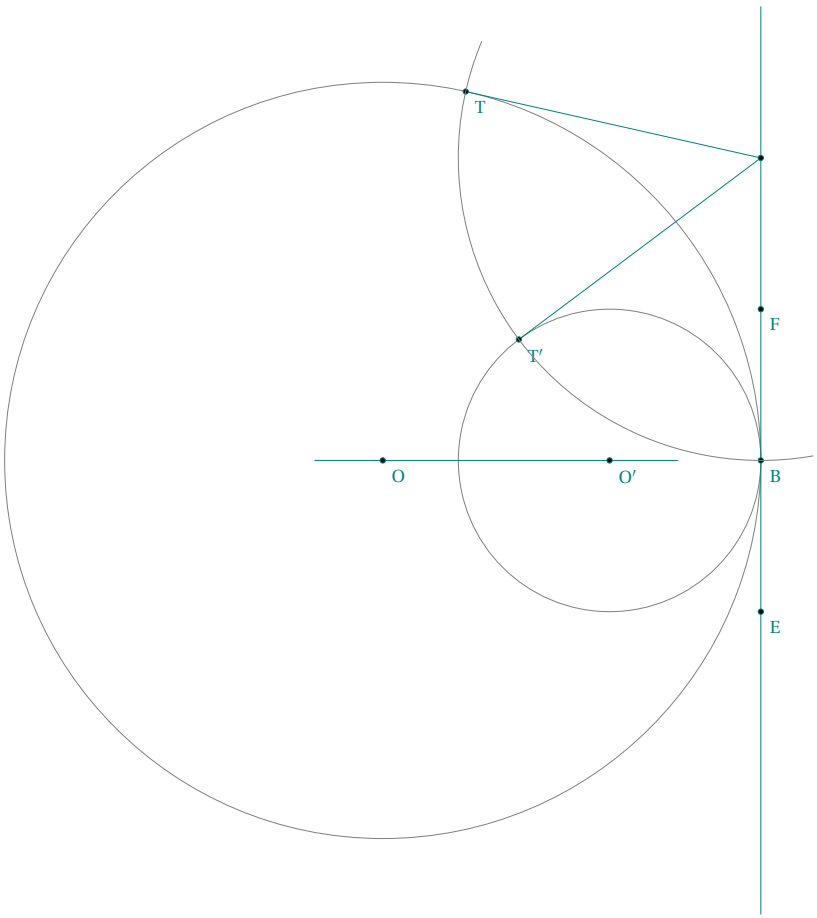
12.33 Radical axis v3

```
\begin{tkzelements}
   z.O     = point : new (0,0)
   z.B     = point : new (4,0)
   z.Op    = point : new (6,0)
   C.OB    = circle :    new (z.O,z.B)
   C.OpB   = circle :    new (z.Op,z.B)
   L.EF    = C.OB : radical_axis (C.OpB)
   z.E,z.F = get_points(L.EF)
   z.M     = L.EF : point (2)
   _,L     = C.OB : tangent_from (z.M)
   _,z.T   = get_points (L)
   L,_     = C.OpB : tangent_from (z.M)
   _,z.Tp  = get_points (L)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(O,B O',B)
   \tkzDrawSegments(M,T M,T')
   \tkzDrawLine[add=.5 and 1](E,F)
   \tkzDrawLine[add=.5 and .5](O,O')
   \tkzDrawPoints(O,B,O',E,F,M,T,T')
   \tkzLabelPoints(O,O',B,E,F,T,T')
   \tkzDrawArc(M,T')(T)
\end{tikzpicture}
```



12.34 Radical axis v4

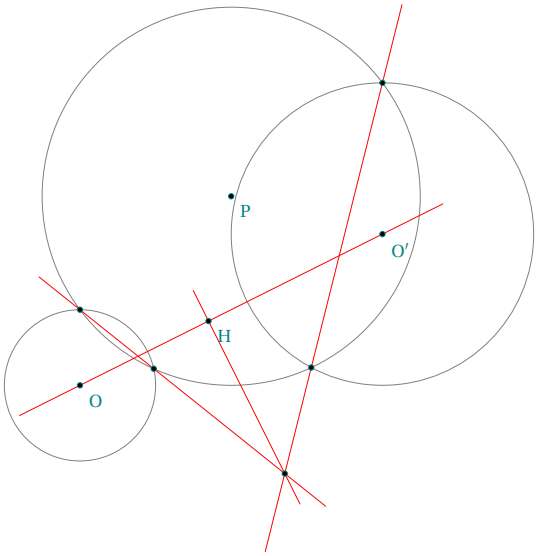
```
\begin{tkzelements}
   z.O     = point : new (0,0)
   z.B     = point : new (5,0)
   z.Op    = point : new (3,0)
   C.OB    = circle :    new (z.O,z.B)
   C.OpB   = circle :    new (z.Op,z.B)
   L.EF    = C.OB : radical_axis (C.OpB)
   z.E,z.F = get_points(L.EF)
   z.M     = L.EF : point (1.5)
   L,_     = C.OB : tangent_from (z.M)
   _,z.T   = get_points (L)
   L,_     = C.OpB : tangent_from (z.M)
   _,z.Tp  = get_points (L)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(O,B O',B)
   \tkzDrawSegments(M,T M,T')
   \tkzDrawLine[add=1 and 1](E,F)
   \tkzDrawLine[add=.3 and .3](O,O')
   \tkzDrawPoints(O,O',B,E,F,T,T',M)
   \tkzLabelPoints[below right](O,O',B,E,F,T,T')
   \tkzDrawArc(M,T)(B)
\end{tikzpicture}
```



12.35 Radical center

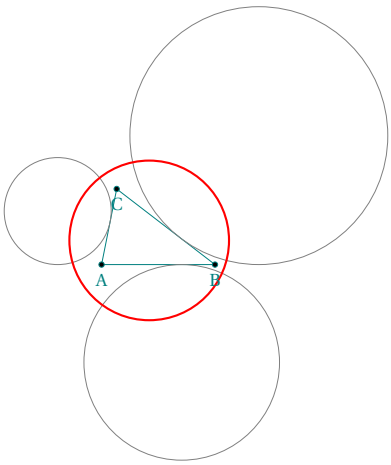
```
\begin{tkzelements}
   z.O      = point : new (0,0)
   z.x      = point : new (1,0)
   z.y      = point : new (4,0)
   z.z      = point : new (2,0)
   z.Op     = point : new (4,2)
   z.P      = point : new (2,2.5)
   C.Ox     = circle :    new (z.O,z.x)
   C.Pz     = circle :    new (z.P,z.z)
   C.Opy    = circle :    new (z.Op,z.y)
   z.ap,z.a = intersection (C.Ox,C.Pz)
   z.bp,z.b = intersection (C.Opy,C.Pz)
   L.aap    = line : new (z.a,z.ap)
   L.bbp    = line : new (z.b,z.bp)
   z.X      = intersection (L.aap,L.bbp)
-- or z.X   = radical_center(C.Ox,C.Pz,C.Opy)
   L.OOp    = line : new (z.O,z.Op)
   z.H      = L.OOp : projection (z.X)
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(O,a O',b P,z)
```

```
   \tkzDrawLines[red](a,X b',X H,X O,O')
   \tkzDrawPoints(O,O',P,a,a',b,b',X,H)
   \tkzLabelPoints[below right](O,O',P,H)
\end{tikzpicture}
```



12.36 Radical circle

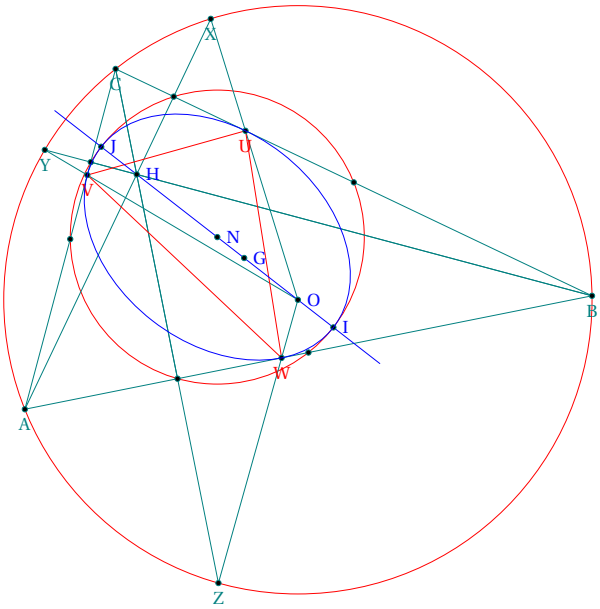
```
\begin{tkzelements}
   scale       = .25
   z.A         = point: new (0,0)
   z.B         = point: new (6,0)
   z.C         = point: new (0.8,4)
   T.ABC       = triangle : new ( z.A,z.B,z.C )
   C.exa       = T.ABC : ex_circle ()
   z.I_a,z.Xa  = get_points (C.exa)
   C.exb       = T.ABC : ex_circle (1)
   z.I_b,z.Xb  = get_points (C.exb)
   C.exc       = T.ABC : ex_circle (2)
   z.I_c,z.Xc  = get_points (C.exc)
   C.ortho     = radical_circle (C.exa,C.exb,C.exc)
   z.w,z.a     = get_points (C.ortho)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon(A,B,C)
   \tkzDrawCircles(I_a,Xa I_b,Xb I_c,Xc)
   \tkzDrawCircles[red,thick](w,a)
   \tkzDrawPoints(A,B,C)
   \tkzLabelPoints(A,B,C)
\end{tikzpicture}
```



12.37 Hexagram

```
\begin{tkzelements}
   z.A               = point: new (0 , 0)
   z.B               = point: new (5 , 1)
   L.AB              = line : new (z.A,z.B)
   z.C               = point: new (.8 , 3)
   T.ABC             = triangle: new (z.A,z.B,z.C)
   z.N               = T.ABC.eulercenter
   z.G               = T.ABC.centroid
   z.O               = T.ABC.circumcenter
   z.H               = T.ABC.orthocenter
   z.Ma,z.Mb,z.Mc    = get_points (T.ABC : medial ())
   z.Ha,z.Hb,z.Hc    = get_points (T.ABC : orthic ())
   z.Ea,z.Eb,z.Ec    = get_points (T.ABC: extouch())
   L.euler           = T.ABC : euler_line ()
   C.circum          = T.ABC : circum_circle ()
   C.euler           = T.ABC : euler_circle ()
   z.I,z.J           = intersection (L.euler,C.euler)
   E                 = ellipse: foci (z.H,z.O,z.I)
   a                 = E.Rx
   b                 = E.Ry
   ang               = math.deg(E.slope)
   L.AH              = line: new (z.A,z.H)
   L.BH              = line: new (z.B,z.H)
   L.CH              = line: new (z.C,z.H)
   z.X               = intersection (L.AH,C.circum)
   _,z.Y             = intersection (L.BH,C.circum)
   _,z.Z             = intersection (L.CH,C.circum)
   L.BC              = line: new (z.B,z.C)
   L.XO              = line: new (z.X,z.O)
   L.YO              = line: new (z.Y,z.O)
   L.ZO              = line: new (z.Z,z.O)
   z.x               = intersection (L.BC,L.XO)
   z.U               = intersection (L.XO,E)
  _,z.V              = intersection (L.YO,E)
  _,z.W              = intersection (L.ZO,E)
 \end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
```

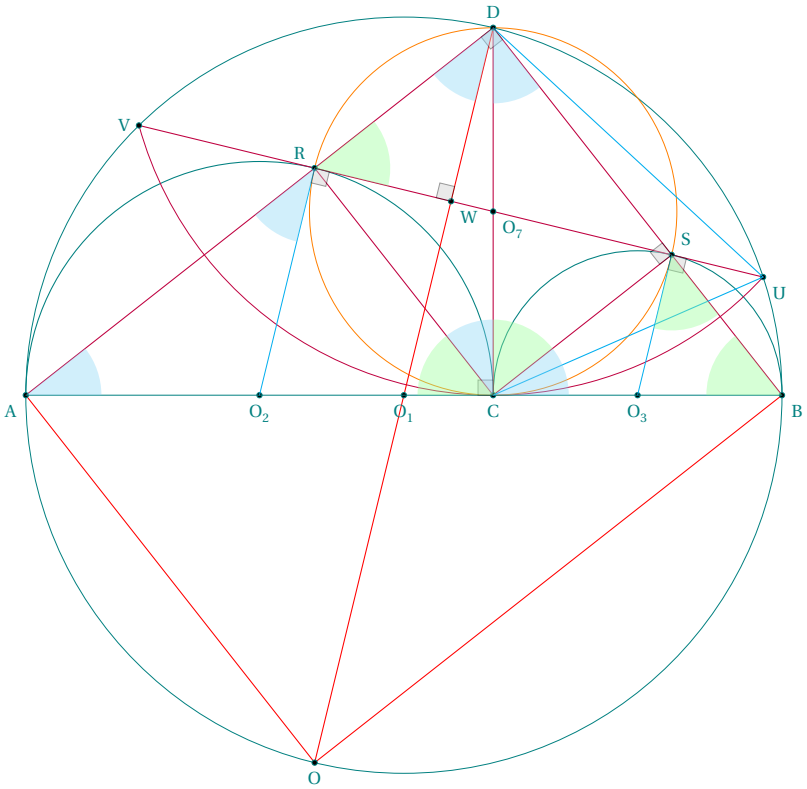
```
   \tkzDrawPolygon(A,B,C)
   \tkzDrawCircles[red](N,Ma O,A)
   \tkzDrawSegments(A,X B,Y C,Z B,Hb C,Hc X,O Y,O Z,O)
   \tkzDrawPolygon[red](U,V,W)
   \tkzLabelPoints[red](U,V,W)
   \tkzLabelPoints(A,B,C,X,Y,Z)
   \tkzDrawLine[blue](I,J)
   \tkzLabelPoints[blue,right](O,N,G,H,I,J)
   \tkzDrawPoints(I,J,U,V,W)
   \tkzDrawPoints(A,B,C,N,G,H,O,X,Y,Z,Ma,Mb,Mc,Ha,Hb,Hc)
   \tkzDrawEllipse[blue](N,\tkzUseLua{a},\tkzUseLua{b},\tkzUseLua{ang})
\end{tikzpicture}
```



12.38 Gold Arbelos properties

```
\begin{tkzelements}
   z.A      = point : new(0,0)
   z.B      = point : new(10,0)
   z.C      = gold_segment_ (z.A,z.B)
   L.AB     = line:new (z.A,z.B)
   z.O_1    = L.AB.mid
   L.AC     = line:new (z.A,z.C)
   z.O_2    = L.AC.mid
   L.CB     = line:new (z.C,z.B)
   z.O_3    = L.CB.mid
   C1       = circle:new (z.O_1,z.B)
   C2       = circle:new (z.O_2,z.C)
   C3       = circle:new (z.O_3,z.B)
   z.Q      = C2.north
   z.P      = C3.north
   L1       = line:new (z.O_2,z.O_3)
   z.M_0    = L1:harmonic_ext (z.C)
   L2       = line:new (z.O_1,z.O_2)
   z.M_1    = L2:harmonic_int (z.A)
   L3       = line:new (z.O_1,z.O_3)
```

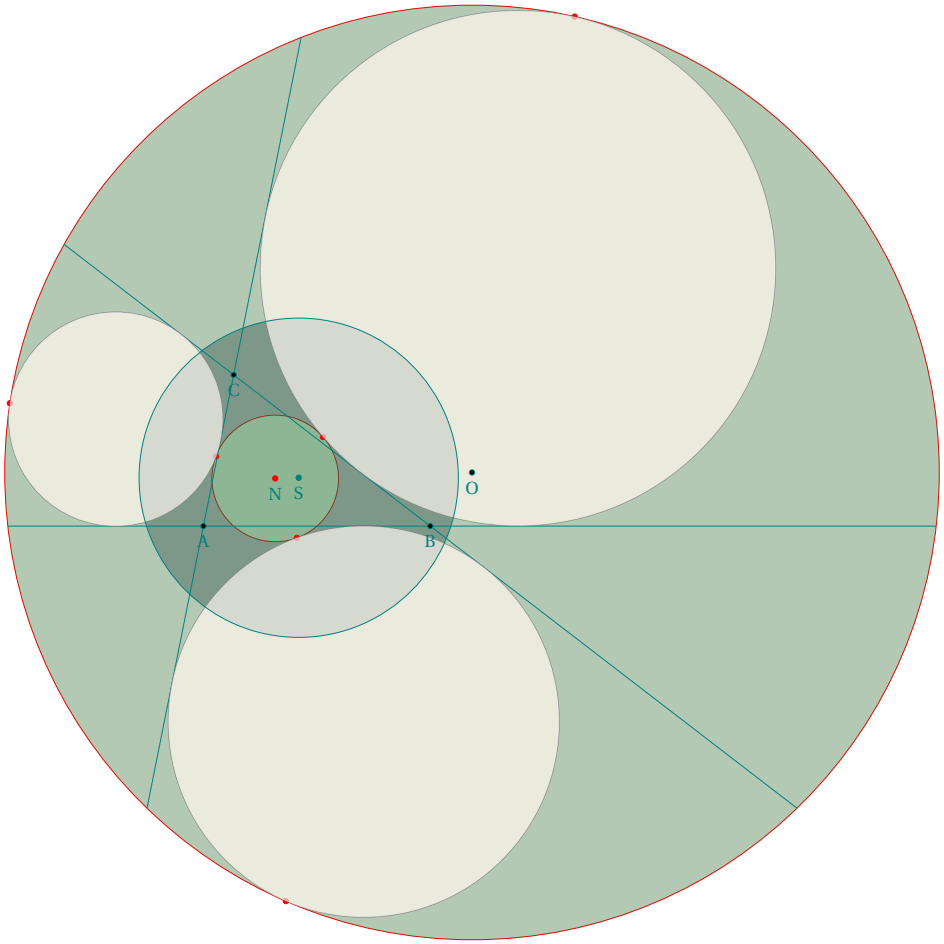
```
   z.M_2    = L3:harmonic_int (z.B)
   Lbq      = line:new (z.B,z.Q)
   Lap      = line:new (z.A,z.P)
   z.S      = intersection (Lbq,Lap)
   z.x      = z.C: north ()
   L        = line : new (z.C,z.x)
   z.D,_    = intersection (L,C1)
   L.CD     = line :new (z.C,z.D)
   z.O_7    = L.CD.mid
   C.DC     = circle: new (z.D,z.C)
   z.U,z.V  = intersection (C.DC,C1)
   L.UV     = line :new (z.U,z.V)
   z.R ,z.S = L.UV : set_projection (z.O_2,z.O_3)
   L.O1D    = line : new (z.O_1,z.D)
   z.W      = intersection (L.UV,L.O1D)
   z.O      = C.DC : inversion (z.W)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles[teal](O_1,B)
   \tkzDrawSemiCircles[thin,teal](O_2,C O_3,B)
   \tkzDrawArc[purple,delta=0](D,V)(U)
   \tkzDrawCircle[new](O_7,C)
   \tkzDrawSegments[thin,purple](A,D D,B C,R C,S C,D U,V)
   \tkzDrawSegments[thin,red](O,D A,O O,B)
   \tkzDrawPoints(A,B,C,D,O_7) %,
   \tkzDrawPoints(O_1,O_2,O_3,U,V,R,S,W,O)
   \tkzDrawSegments[cyan](O_3,S O_2,R)
   \tkzDrawSegments[very thin](A,B)
   \tkzDrawSegments[cyan,thin](C,U U,D)
   \tkzMarkRightAngles[size=.2,fill=gray!40,opacity=.4](D,C,A A,D,B
     D,S,C D,W,V O_3,S,U O_2,R,U)
   \tkzFillAngles[cyan!40,opacity=.4](B,A,D A,D,O_1
     C,D,B D,C,R B,C,S A,R,O_2)
   \tkzFillAngles[green!40,opacity=.4](S,C,D W,R,D
     D,B,C R,C,A O_3,S,B)
   \tkzLabelPoints[below](C,O_2,O_3,O_1)
   \tkzLabelPoints[above](D)
   \tkzLabelPoints[below](O)
   \tkzLabelPoints[below left](A)
   \tkzLabelPoints[above left](R)
   \tkzLabelPoints[above right](S)
   \tkzLabelPoints[left](V)
   \tkzLabelPoints[below right](B,U,W,O_7)
\end{tikzpicture}
```



12.39 Apollonius circle v1 with inversion

```
\begin{tkzelements}
   scale             = .7
   z.A               = point: new (0,0)
   z.B               = point: new (6,0)
   z.C               = point: new (0.8,4)
   T.ABC             = triangle : new ( z.A,z.B,z.C )
   z.N               = T.ABC.eulercenter
   z.Ea,z.Eb,z.Ec    = get_points ( T.ABC : feuerbach () )
   z.Ja,z.Jb,z.Jc    = get_points ( T.ABC : excentral () )
   z.S               = T.ABC : spieker_center ()
   C.JaEa            = circle : new (z.Ja,z.Ea)
   C.ortho           = circle : radius (z.S,math.sqrt(C.JaEa : power (z.S) ))
   z.a               = C.ortho.south
   C.euler           = T.ABC: euler_circle ()
   C.apo             = C.ortho : inversion (C.euler)
   z.O               = C.apo.center
   z.xa,z.xb,z.xc    = C.ortho : set_inversion (z.Ea,z.Eb,z.Ec)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
\tkzDrawCircles[red](O,xa N,Ea)
\tkzFillCircles[green!30!black,opacity=.3](O,xa)
\tkzFillCircles[yellow!30,opacity=.7](Ja,Ea Jb,Eb Jc,Ec)
\tkzFillCircles[teal!30!black,opacity=.3](S,a)
\tkzFillCircles[green!30,opacity=.3](N,Ea)
\tkzDrawPoints[red](Ea,Eb,Ec,xa,xb,xc,N)
```

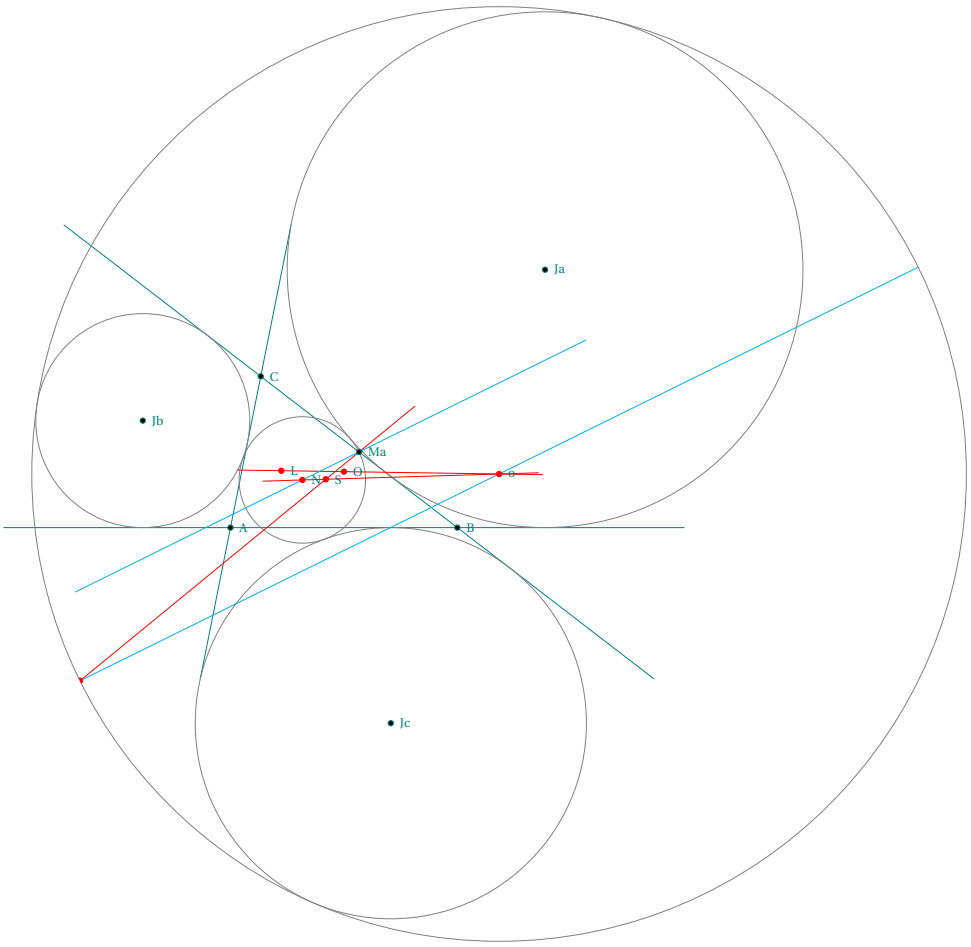
```
\tkzClipCircle(O,xa)
\tkzDrawLines[add=3 and 3](A,B A,C B,C)
\tkzDrawCircles(Ja,Ea Jb,Eb Jc,Ec)
\tkzFillCircles[lightgray!30,opacity=.7](Ja,Ea Jb,Eb Jc,Ec)
\tkzDrawCircles[teal](S,a)
\tkzDrawPoints(A,B,C,O)
\tkzDrawPoints[teal](S)
\tkzLabelPoints(A,B,C,O,S,N)
\end{tikzpicture}
```



12.40 Apollonius circle v2

```
\begin{tkzelements}
   scale       = .5
   z.A         = point: new (0,0)
   z.B         = point: new (6,0)
   z.C         = point: new (0.8,4)
   T.ABC       = triangle: new(z.A,z.B,z.C)
   z.O         = T.ABC.circumcenter
   z.H         = T.ABC.orthocenter
   z.G         = T.ABC.centroid
   z.L         = T.ABC: lemoine_point ()
   z.S         = T.ABC: spieker_center ()
   C.euler     = T.ABC: euler_circle ()
```

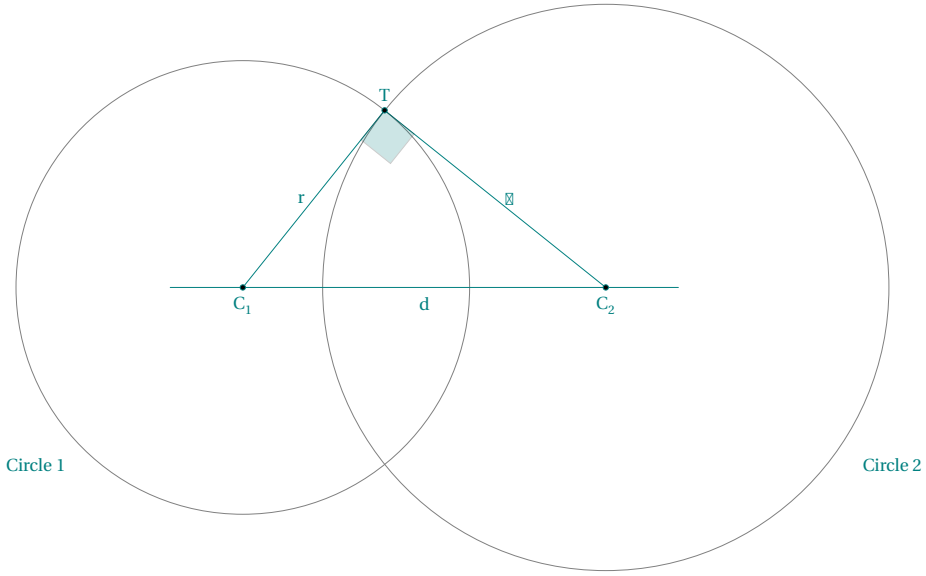
```
   z.N,z.Ma    = get_points (C.euler)
   C.exA       = T.ABC : ex_circle ()
   z.Ja,z.Xa   = get_points (C.exA)
   C.exB       = T.ABC : ex_circle (1)
   z.Jb,z.Xb   = get_points (C.exB)
   C.exC       = T.ABC : ex_circle (2)
   z.Jc,z.Xc   = get_points (C.exC)
   L.OL        = line: new (z.O,z.L)
   L.NS        = line: new (z.N,z.S)
   z.o         = intersection (L.OL,L.NS) -- center of Apollonius circle
   L.NMa       = line: new (z.N,z.Ma)
   L.ox        = L.NMa: ll_from (z.o)
   L.MaS       = line: new (z.Ma,z.S)
   z.t         = intersection (L.ox,L.MaS) -- through
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawLines[add=1 and 1](A,B A,C B,C)
   \tkzDrawCircles(Ja,Xa Jb,Xb Jc,Xc o,t N,Ma) %
   \tkzClipCircle(o,t)
   \tkzDrawLines[red](o,L N,o Ma,t)
   \tkzDrawLines[cyan,add=4 and 4](Ma,N o,t)
   \tkzDrawPoints(A,B,C,Ma,Ja,Jb,Jc)
   \tkzDrawPoints[red](N,O,L,S,o,t)
   \tkzLabelPoints[right,font=\tiny](A,B,C,Ja,Jb,Jc,O,N,L,S,Ma,o)
\end{tikzpicture}
```



12.41 Orthogonal circles v1

```
\begin{tkzelements}
   scale    = .6
   z.C_1    = point: new (0,0)
   z.C_2    = point: new (8,0)
   z.A      = point: new (5,0)
   C        = circle: new (z.C_1,z.A)
   z.S,z.T  = get_points (C: orthogonal_from (z.C_2))
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(C_1,T C_2,T)
   \tkzDrawSegments(C_1,T C_2,T)
   \tkzDrawLine(C_1,C_2)
   \tkzMarkRightAngle[fill=teal,%
  opacity=.2,size=1](C_1,T,C_2)
   \tkzDrawPoints(C_1,C_2,T)
   \tkzLabelPoints(C_1,C_2)
   \tkzLabelPoints[above](T)
   \tkzLabelSegment[left](C_1,T){r}
   \tkzLabelSegments[right](C_2,T){\gamma}
   \tkzLabelSegment[below](C_1,C_2){d}
   \tkzLabelCircle[left=10pt](C_1,T)(180){Circle 1}
```

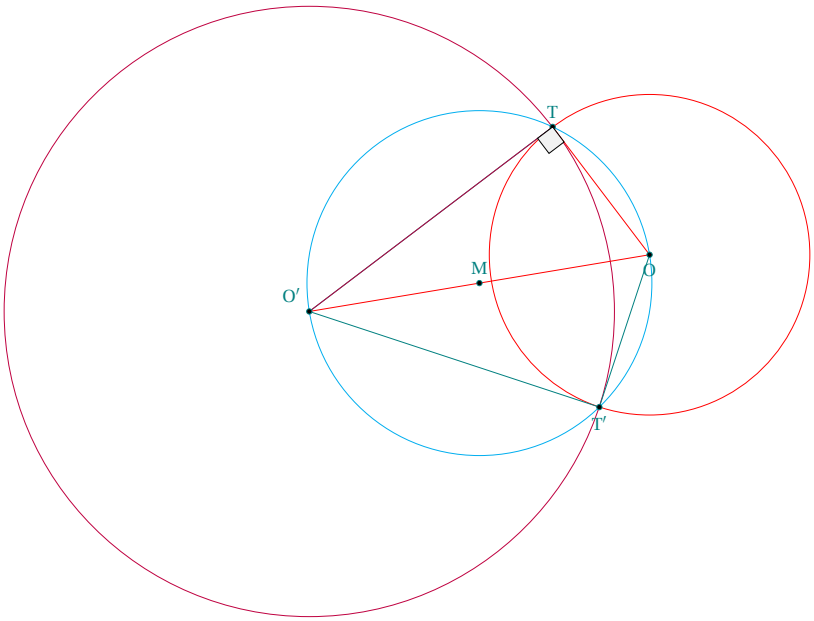
```
   \tkzLabelCircle[right=10pt](C_2,T)(180){Circle 2}
\end{tikzpicture}
```



12.42 Orthogonal circles v2

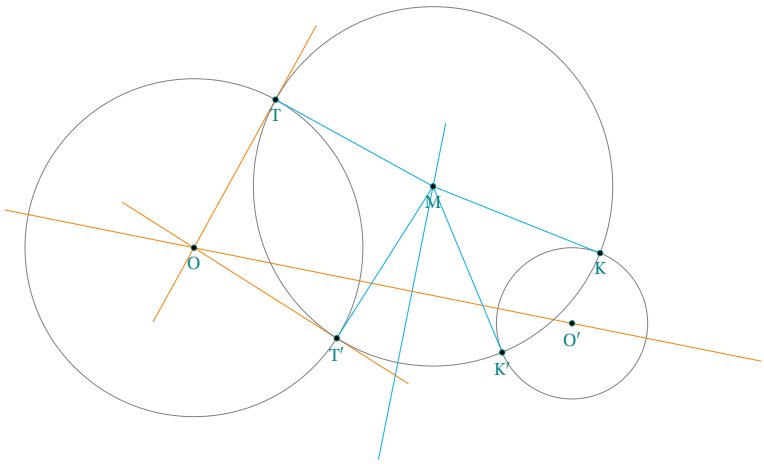
```
\begin{tkzelements}
   scale       = .75
   z.O         = point: new (2,2)
   z.Op        = point: new (-4,1)
   z.P         = point: polar (4,0)
   C.OP        = circle: new (z.O,z.P)
   C.Oz1,C.Oz2 = C.OP : orthogonal_from (z.Op)
   z.z1        = C.Oz1.through
   z.z2        = C.Oz2.through
   L.OP        = line : new (z.O,z.P)
   C.Opz1      = circle: new (z.Op,z.z1)
   L.T,L.Tp    = C.Opz1 : tangent_from (z.O)
   z.T         = L.T.pb
   z.Tp        = L.Tp.pb
   L.OOp       = line : new (z.O,z.Op)
   z.M         = L.OOp.mid
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircle[red](O,P)
   \tkzDrawCircle[purple](O',z1)
   \tkzDrawCircle[cyan](M,T)
   \tkzDrawSegments(O',T O,T' O',T')
   \tkzDrawSegment[purple](O',T)
   \tkzDrawSegments[red](O,T O,O')
   \tkzDrawPoints(O,O',T,T',M)
   \tkzMarkRightAngle[fill=gray!10](O',T,O)
   \tkzLabelPoint[below](O){$O$}
   \tkzLabelPoint[above](T){$T$}
   \tkzLabelPoint[above](M){$M$}
   \tkzLabelPoint[below](T'){$T'$}
```

```
   \tkzLabelPoint[above left](O'){$O'$}
\end{tikzpicture}
```



12.43 Orthogonal circle to two circles

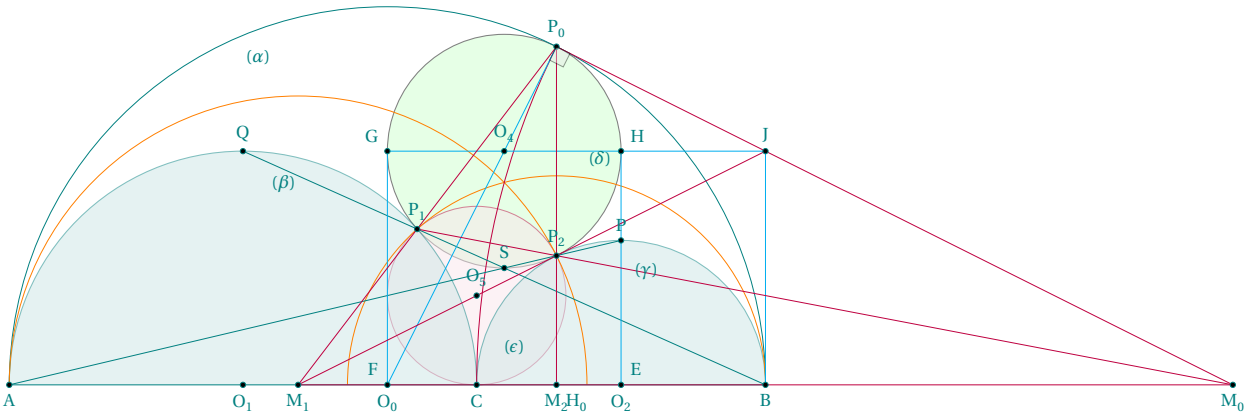
```
\begin{tkzelements}
   z.O          = point :   new (-1,0)
   z.B          = point :   new (0,2)
   z.Op         = point :   new (4,-1)
   z.D          = point :   new (4,0)
   C.OB         = circle :  new (z.O,z.B)
   C.OpD        = circle :  new (z.Op,z.D)
   z.E,z.F      = get_points (C.OB : radical_axis (C.OpD))
   L.EF         = line : new (z.E,z.F)
   z.M          = L.EF : point (2.25)
   L.T,L.Tp     = C.OB : tangent_from (z.M)
   L.K,L.Kp     = C.OpD : tangent_from (z.M)
   z.T          = L.T.pb
   z.K          = L.K.pb
   z.Tp         = L.Tp.pb
   z.Kp         = L.Kp.pb
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(O,B O',D)
   \tkzDrawLine[add=1 and 2,cyan](E,F)
   \tkzDrawLines[add=.5 and .5,orange](O,O' O,T O,T')
   \tkzDrawSegments[cyan](M,T M,T' M,K M,K')
   \tkzDrawCircle(M,T)
   \tkzDrawPoints(O,O',T,M,T',K,K')
   \tkzLabelPoints(O,O',T,T',M,K,K')
\end{tikzpicture}
```



12.44 Midcircles

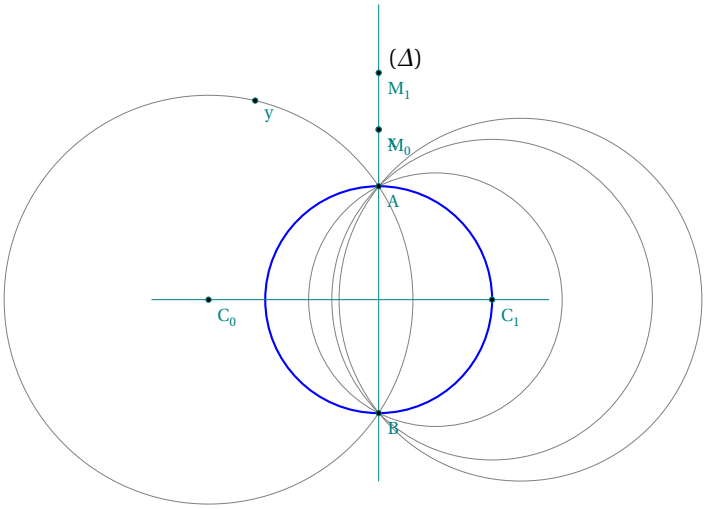
```
\begin{tkzelements}
   z.A      = point: new (0 , 0)
   z.B      = point: new (10 , 0)
   L.AB     = line : new (z.A,z.B)
   z.C      = L.AB: gold_ratio ()
   L.AC     = line : new (z.A,z.C)
   L.CB     = line : new (z.C,z.B)
   z.O_0    = L.AB.mid
   z.O_1    = L.AC.mid
   z.O_2    = L.CB.mid
   C.O0B    = circle : new (z.O_0,z.B)
   C.O1C    = circle : new (z.O_1,z.C)
   C.O2C    = circle : new (z.O_2,z.B)
   z.Q      = C.O1C : midarc (z.C,z.A)
   z.P      = C.O2C : midarc (z.B,z.C)
   L.O1O2   = line : new (z.O_1,z.O_2)
   L.O0O1   = line : new (z.O_0,z.O_1)
   L.O0O2   = line : new (z.O_0,z.O_2)
   z.M_0    = L.O1O2 : harmonic_ext (z.C)
   z.M_1    = L.O0O1 : harmonic_int (z.A)
   z.M_2    = L.O0O2 : harmonic_int (z.B)
   L.BQ     = line : new (z.B,z.Q)
   L.AP     = line : new (z.A,z.P)
   z.S      = intersection (L.BQ,L.AP)
   L.CS     = line : new (z.C,z.S)
   C.M1A    = circle : new (z.M_1,z.A)
   C.M2B    = circle : new (z.M_2,z.B)
   z.P_0    = intersection (L.CS,C.O0B)
   z.P_1    = intersection (C.M2B,C.O1C)
   z.P_2    = intersection (C.M1A,C.O2C)
   T.P012   = triangle : new (z.P_0,z.P_1,z.P_2)
   z.O_4    = T.P012.circumcenter
   T.CP12   = triangle : new (z.C,z.P_1,z.P_2)
   z.O_5    = T.CP12.circumcenter
   z.BN     = z.B : north ()
   L.BBN    = line : new (z.B,z.BN)
   L.M1P2   = line : new (z.M_1,z.P_2)
   z.J      = intersection (L.BBN,L.M1P2)
```

```
   L.AP0    = line : new (z.A,z.P_0)
   L.BP0    = line : new (z.B,z.P_0)
   C.O4P0   = circle : new (z.O_4,z.P_0)
   _,z.G    = intersection (L.AP0,C.O4P0)
   z.H      = intersection (L.BP0,C.O4P0)
   z.Ap     = z.M_1: symmetry (z.A)
   z.H_4,z.F,z.E,z.H_0 = L.AB : set_projection (z.O_4,z.G,z.H,z.P_0)
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircle[thin,fill=green!10](O_4,P_0)
   \tkzDrawCircle[purple,fill=purple!10,opacity=.5](O_5,C)
   \tkzDrawSemiCircles[teal](O_0,B)
   \tkzDrawSemiCircles[thin,teal,fill=teal!20,opacity=.5](O_1,C O_2,B)
   \tkzDrawSemiCircles[new](M_2,B)
   \tkzDrawSemiCircles[new](M_1,A')
   \tkzDrawArc[purple,delta=0](M_0,P_0)(C)
   \tkzDrawSegments[very thin](A,B A,P B,Q)
   \tkzDrawSegments[step 1](O_0,P_0 B,J G,J G,O_0 H,O_2)
   \tkzDrawSegments[ultra thin,purple](M_1,P_0 M_2,P_0 M_1,M_0 M_0,P_1 M_0,P_0 M_1,J)
   \tkzDrawPoints(A,B,C,P_0,P_2,P_1,M_0,M_1,M_2,J,P,Q,S)
   \tkzDrawPoints(O_0,O_1,O_2,O_4,O_5,G,H)
   \tkzMarkRightAngle[size=.2,fill=gray!20,opacity=.4](O_0,P_0,M_0)
   \tkzLabelPoints[below](A,B,C,M_0,M_1,M_2,O_1,O_2,O_0)
   \tkzLabelPoints[above](P_0,O_5,O_4)
   \tkzLabelPoints[above](P_1,J)
   \tkzLabelPoints[above](P_2,P,Q,S)
   \tkzLabelPoints[above right](H,E)
   \tkzLabelPoints[above left](F,G)
   \tkzLabelPoints[below right](H_0)
   \tkzLabelCircle[below=4pt,font=\scriptsize](O_1,C)(80){$(\beta)$}
   \tkzLabelCircle[below=4pt,font=\scriptsize](O_2,B)(80){$(\gamma)$}
   \tkzLabelCircle[below=4pt,font=\scriptsize](O_0,B)(110){$(\alpha)$}
   \tkzLabelCircle[left,font=\scriptsize](O_4,P_2)(60){$(\delta)$}
   \tkzLabelCircle[above left,font=\scriptsize](O_5,C)(40){$(\epsilon)$}
   \end{tikzpicture}
```



12.45 Pencil v1

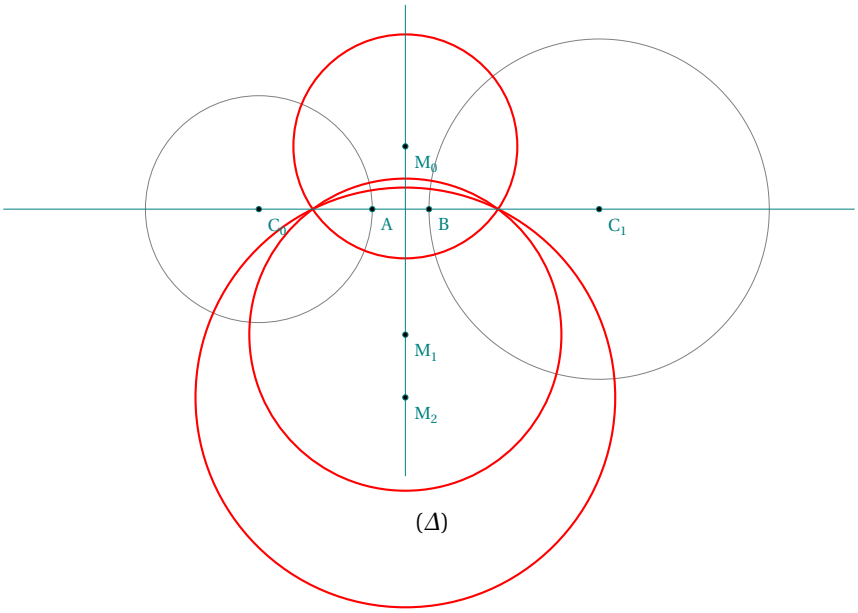
```
\begin{tkzelements}
   scale       = .75
   z.A         = point : new (0,2)
   z.B         = point : new (0,-2)
   z.C_0       = point : new (-3,0)
   z.C_1       = point : new (2,0)
   z.C_3       = point : new (2.5,0)
   z.C_5       = point : new (1,0)
   L.BA        = line : new (z.B,z.A)
   z.M_0       = L.BA : point (1.25)
   z.M_1       = L.BA : point (1.5)
   C.C0A       = circle :    new (z.C_0,z.A)
   z.x,z.y     = get_points (C.C0A : orthogonal_from (z.M_0))
   z.xp,z.yp   = get_points (C.C0A : orthogonal_from (z.M_1))
   z.O         = L.BA.mid
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(C_0,A C_1,A C_3,A C_5,A)
   \tkzDrawCircles[thick,color=red](M_0,x M_1,x')
   \tkzDrawCircles[thick,color=blue](O,A)
   \tkzDrawLines(C_0,C_1 B,M_1)
   \tkzDrawPoints(A,B,C_0,C_1,M_0,M_1,x,y)
   \tkzLabelPoints[below right](A,B,C_0,C_1,M_0,M_1,x,y)
   \tkzLabelLine[pos=1.25,right]( M_0,M_1){$(\Delta)$}
\end{tikzpicture}
```



12.46 Pencil v2

```
\begin{tkzelements}
   scale=.75
   z.A     = point : new (0,0)
   z.B     = point : new (1,0)
   z.C_0   = point : new (-2,0)
   z.C_1   = point : new (4,0)
   C.C0A   = circle : new (z.C_0,z.A)
   C.C1B   = circle : new (z.C_1,z.B)
   L.EF    = C.C0A : radical_axis (C.C1B)
```

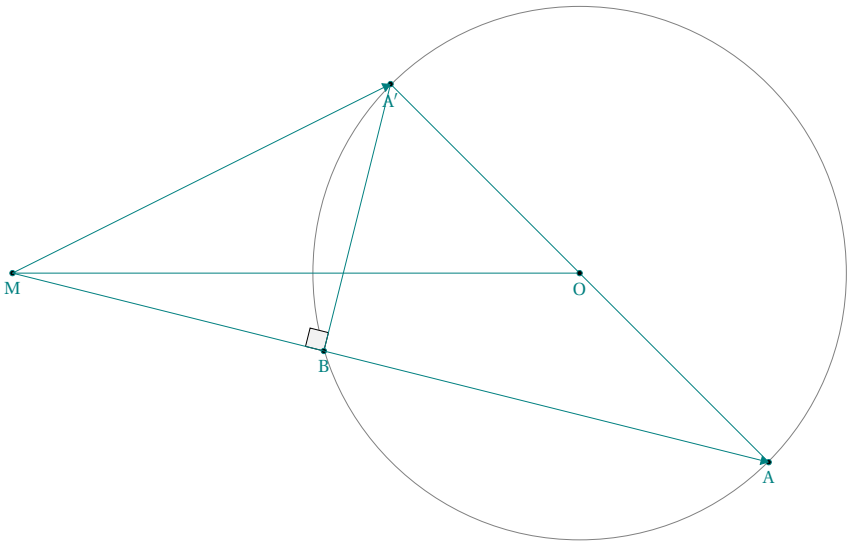
```
   z.M_0   = L.EF : point (2)
   z.M_1   = L.EF : point (-1)
   z.M_2   = L.EF : point (-2)
   C.orth0     = C.C0A : orthogonal_from (z.M_0)
   C.orth1     = C.C0A : orthogonal_from (z.M_1)
   C.orth2     = C.C0A : orthogonal_from (z.M_2)
   z.u         = C.orth0.through
   z.v         = C.orth1.through
   z.t         = C.orth2.through
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(C_0,A C_1,B)
   \tkzDrawCircles[thick,color=red](M_0,u M_1,v M_2,t)
   \tkzDrawLines[add= .75 and .75](C_0,C_1 M_0,M_1)
   \tkzDrawPoints(A,B,C_0,C_1,M_0,M_1,M_2)
   \tkzLabelPoints[below right](A,B,C_0,C_1,M_0,M_1,M_2)
   \tkzLabelLine[pos=2,right]( M_0,M_1){$(\Delta)$}
\end{tikzpicture}
```



12.47 Power v1

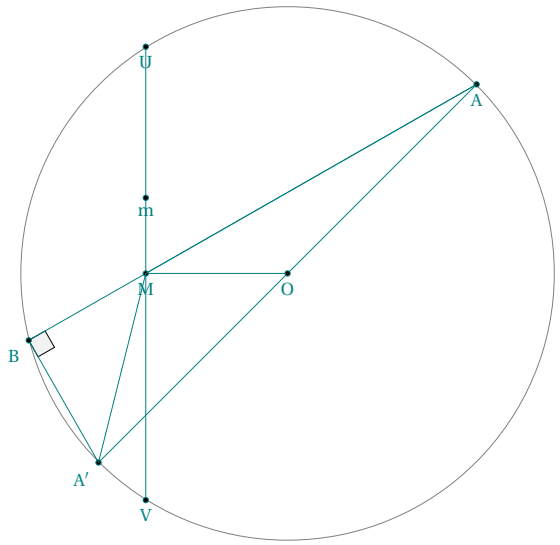
```
\begin{tkzelements}
   z.O     = point : new (0,0)
   z.A     = point : new (2,-2)
   z.M     = point : new (-6,0)
   L.AM    = line : new (z.A,z.M)
   C.OA    = circle :    new (z.O,z.A)
   z.Ap    = C.OA : antipode (z.A)
   z.B     = intersection (L.AM, C.OA)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircle(O,A)
   \tkzMarkRightAngle[fill=gray!10](A',B,M)
```

```
   \tkzDrawSegments(M,O A,A' A',B)
   \tkzDrawPoints(O,A,A',M,B)
   \tkzLabelPoints(O,A,A',M,B)
   \tkzDrawSegments[-Triangle](M,A M,A')
\end{tikzpicture}
```



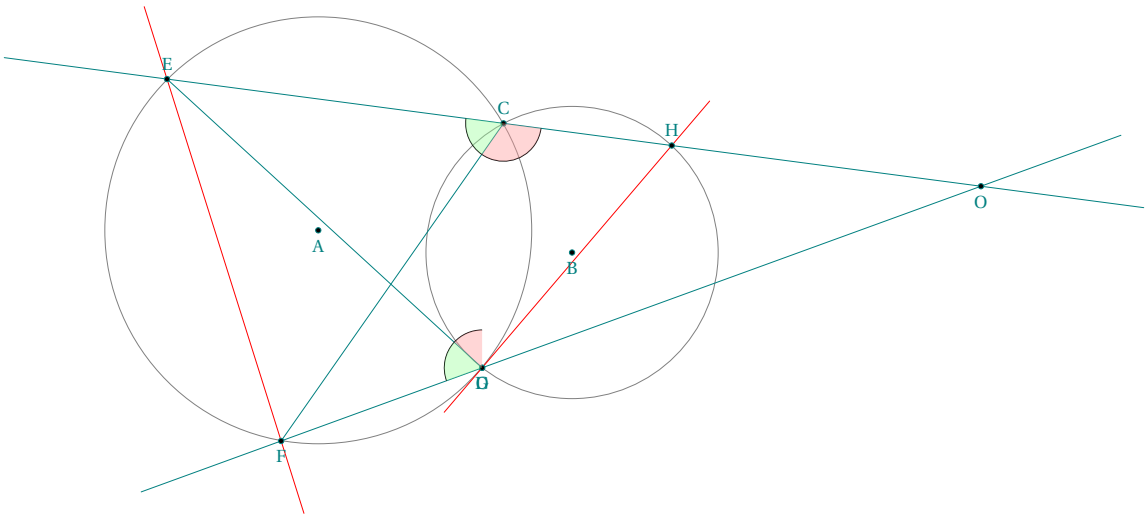
12.48 Power v2

```
\begin{tkzelements}
   z.O     = point : new (0,0)
   z.A     = point : new (2,2)
   z.M     = point : new (-1.5,0)
   L.AM    = line : new (z.A,z.M)
   C.OA    = circle :    new (z.O,z.A)
   z.Ap    = C.OA : antipode (z.A)
   _,z.B   = intersection (L.AM, C.OA)
   z.m     = z.M : north(1)
   L.mM    = line : new (z.m,z.M)
   z.U,z.V = intersection (L.mM,C.OA)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircle(O,A)
   \tkzMarkRightAngle[fill=gray!10](A',B,M)
   \tkzDrawSegments(M,O A,A' A',B A,B U,V)
   \tkzDrawPoints(O,A,A',M,B,U,V,m)
   \tkzLabelPoints(O,A,M,U,V,m)
   \tkzLabelPoints[below left](A',B)
   \tkzDrawSegments(M,A M,A')
\end{tikzpicture}
```



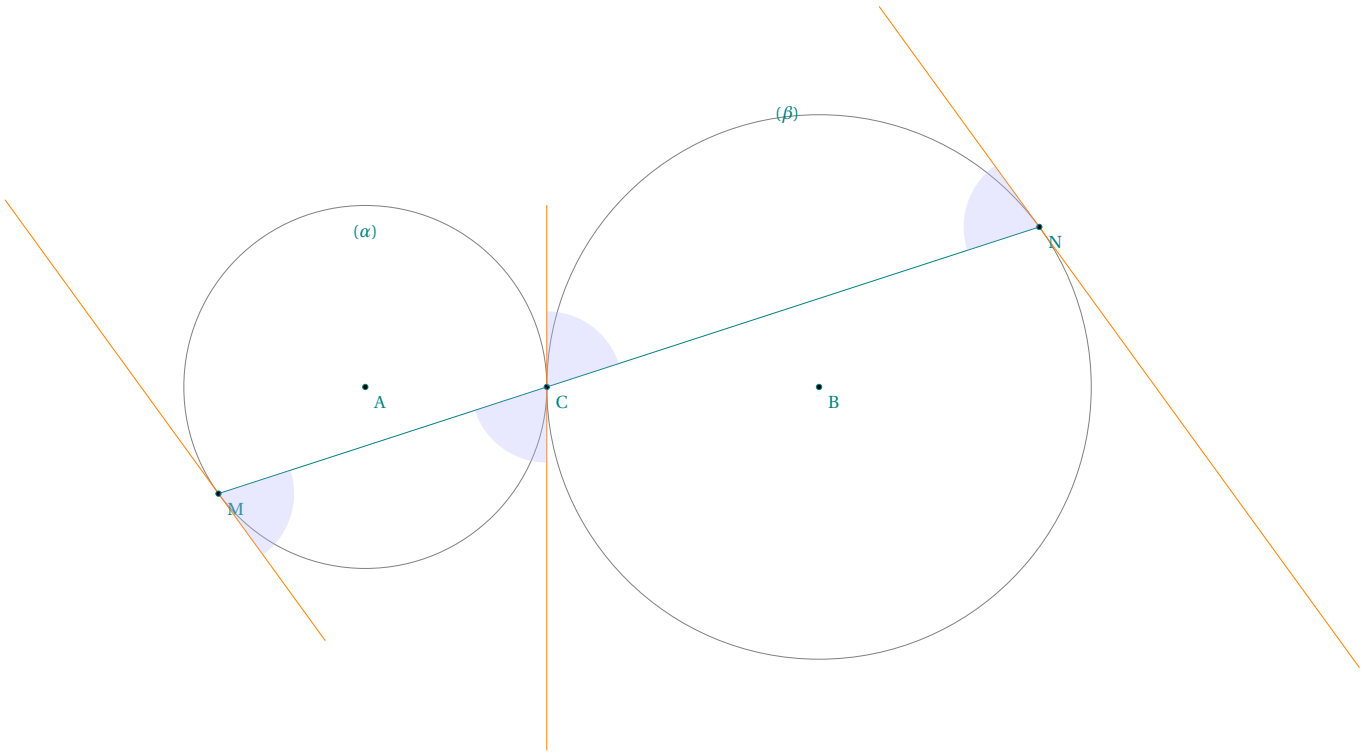
12.49 Reim v1

```
\begin{tkzelements}
   z.A     = point: new (0,0)
   z.E     = point: new (-2,2)
   C.AE    = circle :   new (z.A,z.E)
   z.C     = C.AE : point (math.pi/6)
   z.D     = C.AE : point (-math.pi/180*40)
   z.F     = C.AE : point (-math.pi/180*100)
   L.EC    = line: new (z.E,z.C)
   z.H     = L.EC : point (1.5)
   T.CDH   = triangle : new (z.C,z.D,z.H)
   z.B     = T.CDH.circumcenter
   C.BD    = circle : new (z.B,z.D)
   L.FD    = line: new (z.F,z.D)
   z.G     = intersection (L.FD,C.BD)
   z.O     = intersection (L.EC,L.FD)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(A,E B,H)
   \tkzDrawSegments(E,D C,F)
   \tkzDrawLines(E,O F,O)
   \tkzDrawLines[red](E,F H,G)
   \tkzDrawPoints(A,...,H,O)
   \tkzLabelPoints(A,B,D,F,G,O)
   \tkzLabelPoints[above](E,C,H)
   \tkzMarkAngles[size=.5](E,C,F E,D,F)
   \tkzFillAngles[green!40,opacity=.4,size=.5](E,C,F E,D,F)
   \tkzMarkAngles[size=.5](F,C,H G,D,E)
   \tkzFillAngles[red!40,opacity=.4,size=.5](F,C,H G,D,E)
\end{tikzpicture}
```



12.50 Reim v2

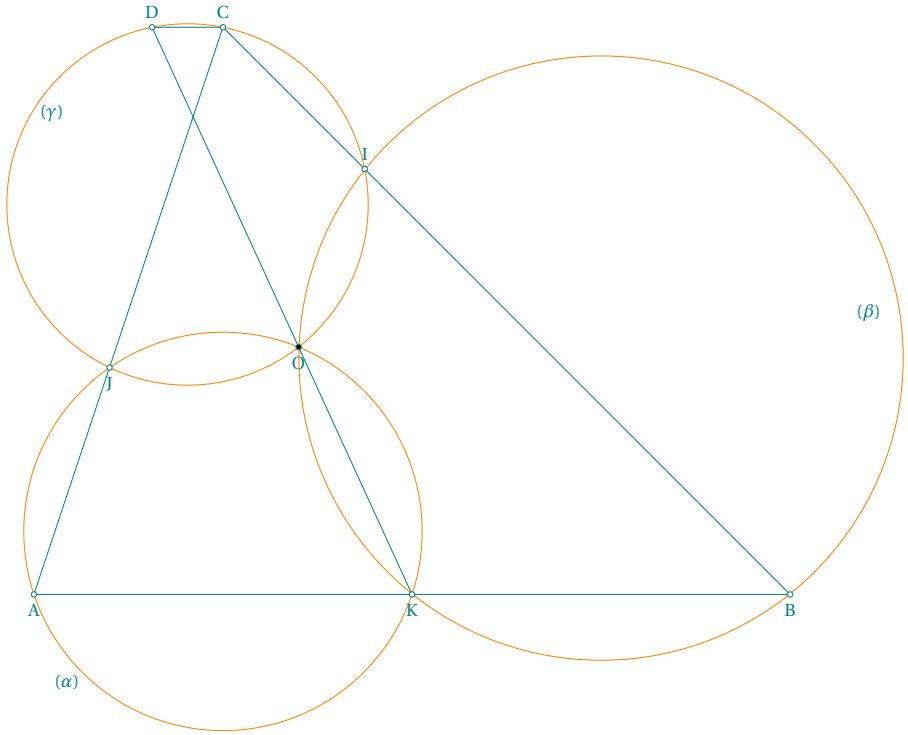
```
\begin{tkzelements}
   scale    = .6
   z.A      = point: new (0,0)
   z.B      = point: new (10,0)
   z.C      = point: new (4,0)
   C.AC     = circle: new (z.A,z.C)
   z.c,z.cp = get_points (C.AC: tangent_at (z.C))
   z.M      = C.AC: point (1.2*math.pi)
   L.MC     = line: new (z.M,z.C)
   C.BC     = circle: new (z.B,z.C)
   z.N      = intersection (L.MC,C.BC)
   z.m,z.mp = get_points (C.AC: tangent_at (z.M))
   z.n,z.np = get_points (C.BC: tangent_at (z.N))
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(A,C B,C)
   \tkzDrawLines[new,add=1 and 1](M,m N,n C,c)
   \tkzDrawSegment(M,N)
   \tkzDrawPoints(A,B,C,M,N)
   \tkzLabelPoints[below right](A,B,C,M,N)
   \tkzFillAngles[blue!30,opacity=.3](m',M,C N,C,c' M,C,c n',N,C)
   \tkzLabelCircle[below=4pt,font=\scriptsize](A,C)(90){$(\alpha)$}
   \tkzLabelCircle[left=4pt,font=\scriptsize](B,C)(-90){$(\beta)$}
\end{tikzpicture}
```



12.51 Reim v3

```
\begin{tkzelements}
   z.A     = point: new (0,0)
   z.B     = point: new (8,0)
   z.C     = point: new (2,6)
   L.AB    = line : new (z.A,z.B)
   L.AC    = line : new (z.A,z.C)
   L.BC    = line : new (z.B,z.C)
   z.I     = L.BC : point (0.75)
   z.J     = L.AC : point (0.4)
   z.K     = L.AB : point (0.5)
   T.AKJ   = triangle : new (z.A,z.K,z.J)
   T.BIK   = triangle : new (z.B,z.I,z.K)
   T.CIJ   = triangle : new (z.C,z.I,z.J)
   z.x     = T.AKJ.circumcenter
   z.y     = T.BIK.circumcenter
   z.z     = T.CIJ.circumcenter
   C.xK    = circle: new (z.x,z.K)
   C.yK    = circle: new (z.y,z.K)
   z.O,_   = intersection (C.xK,C.yK)
   C.zO    = circle: new (z.z,z.O)
   L.KO    = line: new (z.K,z.O)
   z.D     = intersection (L.KO,C.zO)
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawSegments(K,D D,C)
```

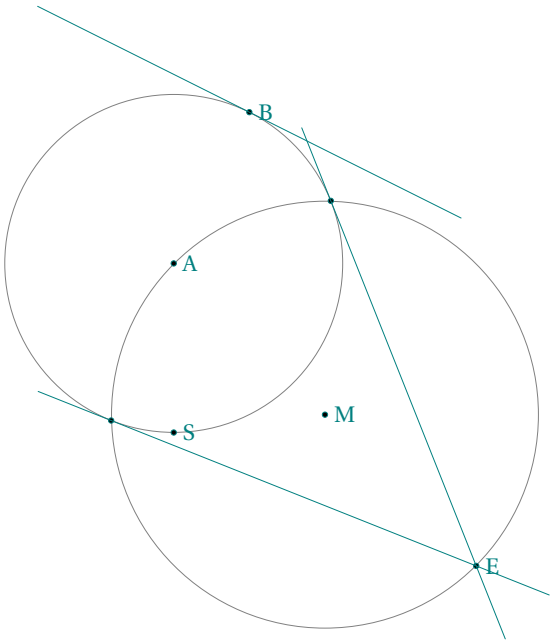
```
   \tkzDrawPolygon[teal](A,B,C)
   \tkzDrawCircles[orange](x,A y,B z,C)
   \tkzDrawPoints[fill=white](A,B,C,I,J,K,D)
   \tkzLabelPoints[below](A,B,J,K,O)
   \tkzLabelPoints[above](C,D,I)
   \tkzDrawPoints[fill=black](O)
   \tkzLabelCircle[below=4pt,font=\scriptsize](x,A)(20){$(\alpha)$}
   \tkzLabelCircle[left=4pt,font=\scriptsize](y,B)(60){$(\beta)$}
   \tkzLabelCircle[below=4pt,font=\scriptsize](z,C)(60){$(\gamma)$}
\end{tikzpicture}
```



12.52 Tangent and circle

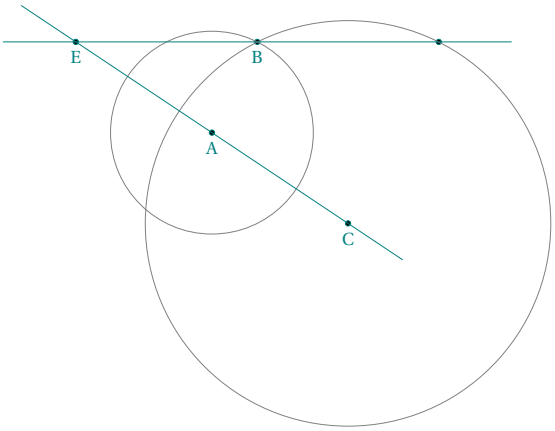
```
\begin{tkzelements}
   z.A          = point:    new (1,0)
   z.B          = point:    new (2,2)
   z.E          = point:    new (5,-4)
   L.AE         = line :    new (z.A,z.E)
   C.AB         = circle:   new (z.A , z.B)
   z.S          = C.AB.south
   z.M          = L.AE.mid
   L.Ti,L.Tj    = C.AB:   tangent_from (z.E)
   z.i          = L.Ti.pb
   z.j          = L.Tj.pb
   z.k,z.l      = get_points (C.AB:  tangent_at (z.B))
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(A,B M,A)
   \tkzDrawPoints(A,B,E,i,j,M,S)
   \tkzDrawLines(E,i E,j k,l)
```

```
   \tkzLabelPoints[right,font=\small](A,B,E,S,M)
\end{tikzpicture}
```



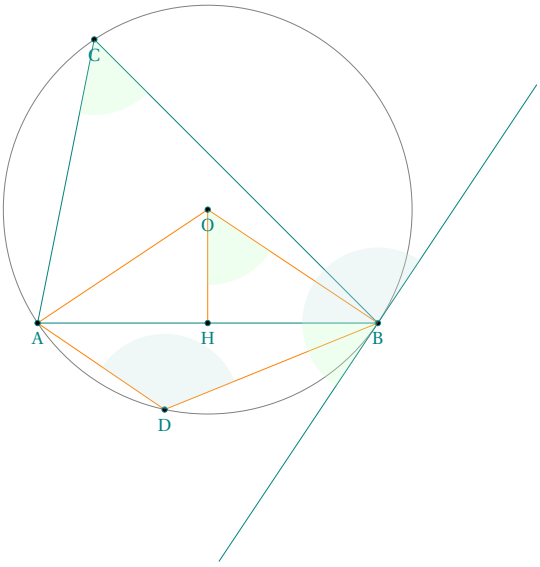
12.53 Homothety

```
\begin{tkzelements}
   z.A      = point:  new (0,0)
   z.B      = point:  new (1,2)
   z.E      = point:  new (-3,2)
   z.C,z.D  = z.E : set_homothety(2,z.A,z.B)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPoints(A,B,C,E,D)
   \tkzLabelPoints(A,B,C,E)
   \tkzDrawCircles(A,B C,D)
   \tkzDrawLines(E,C E,D)
\end{tikzpicture}
```



12.54 Tangent and chord

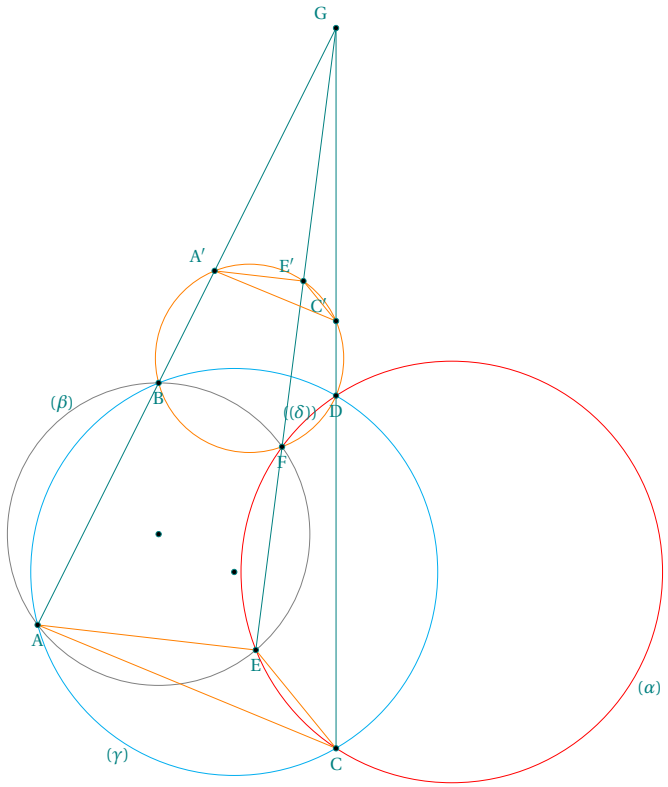
```
\begin{tkzelements}
   scale       = .8
   z.A         = point: new (0 , 0)
   z.B         = point: new (6 , 0)
   z.C         = point: new (1 , 5)
   z.Bp        = point: new (2 , 0)
   T.ABC       = triangle: new (z.A,z.B,z.C)
   L.AB        = line: new (z.A,z.B)
   z.O         = T.ABC.circumcenter
   C.OA        = circle: new (z.O,z.A)
   z.D         = C.OA: point (4.5)
   L.AO        = line: new (z.A,z.O)
   z.b1,z.b2   = get_points (C.OA: tangent_at (z.B))
   z.H         = L.AB: projection (z.O)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircle(O,A)
   \tkzDrawPolygon(A,B,C)
   \tkzDrawSegments[new](A,O B,O O,H A,D D,B)
   \tkzDrawLine(b1,b2)
   \tkzDrawPoints(A,B,C,D,H,O)
   \tkzFillAngles[green!20,opacity=.3](H,O,B A,C,B  A,B,b1)
   \tkzFillAngles[teal!20,opacity=.3](B,D,A b2,B,A)
   \tkzLabelPoints(A,B,C,D,H,O)
\end{tikzpicture}
```



12.55 Three chords

```
\begin{tkzelements}
   z.O      = point: new (0 , 0)
   z.B      = point: new (0 , 2)
   z.P      = point: new (1 , -.5)
   C.OB     = circle : new (z.O,z.B)
```

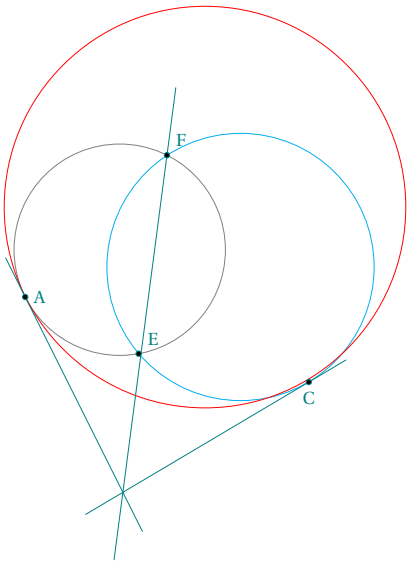
```
   C.PB     = circle : new (z.P,z.B)
   _,z.A    = intersection (C.OB,C.PB)
   z.D      = C.PB: point(math.pi/3)
   z.C      = C.PB: point(-math.pi/3)
   z.E      = C.OB: point(-math.pi/180*50)
   L.AB     = line : new (z.A,z.B)
   L.CD     = line : new (z.C,z.D)
   z.G      = intersection (L.AB,L.CD)
   L.GE     = line : new (z.G,z.E)
   z.F,_    = intersection (L.GE,C.OB)
   T.CDE    = triangle: new (z.C,z.D,z.E)
   T.BFD    = triangle: new (z.B,z.F,z.D)
   z.w      = T.CDE.circumcenter
   z.x      = T.BFD.circumcenter
   L.GB     = line : new (z.G,z.B)
   L.GE     = line : new (z.G,z.E)
   L.GD     = line : new (z.G,z.D)
   C.xB     = circle : new (z.x,z.B)
   C.xF     = circle : new (z.x,z.F)
   C.xD     = circle : new (z.x,z.D)
   z.Ap     = intersection (L.GB,C.xB)
   z.Ep     = intersection (L.GE,C.xF)
   z.Cp     = intersection (L.GD,C.xD)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(O,B)
   \tkzDrawCircles[cyan](P,B)
   \tkzDrawCircles[red](w,E)
   \tkzDrawCircles[new](x,F)
   \tkzDrawSegments(A,G E,G C,G)
   \tkzDrawPolygons[new](A,E,C A',E',C')
   \tkzDrawPoints(A,...,G,A',E',C',O,P)
   \begin{scope}[font=\scriptsize]
   \tkzLabelPoints(A,...,F)
   \tkzLabelPoints[above left](G,A',E',C')
   \tkzLabelCircle[left](O,B)(30){$(\beta)$}
   \tkzLabelCircle[below](P,A)(40){$(\gamma)$}
   \tkzLabelCircle[right](w,C)(90){$(\alpha)$}
   \tkzLabelCircle[left](x,B)(-230){$((\delta))$}
   \end{scope}
\end{tikzpicture}
```



12.56 Three tangents

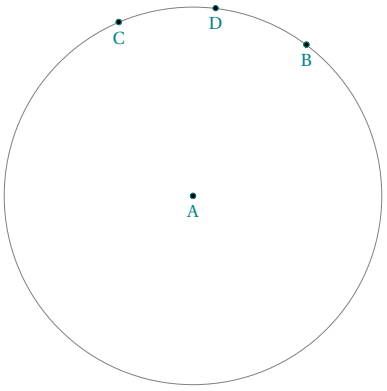
```
\begin{tkzelements}
   z.A    = point: new (-1 , 0)
   z.C    = point: new (4 , -1.5)
   z.E    = point: new (1 , -1)
   z.F    = point: new (1.5 , 2.5)
   T.AEF  = triangle : new (z.A,z.E,z.F)
   T.CEF  = triangle : new (z.C,z.E,z.F)
   z.w    = T.AEF.circumcenter
   z.x    = T.CEF.circumcenter
   C.wE   = circle : new (z.w,z.E)
   C.xE   = circle : new (z.x,z.E)
   L.Aw   = line : new (z.A,z.w)
   L.Cx   = line : new (z.C,z.x)
   z.G    = intersection (L.Aw,L.Cx)
   L.TA   = C.wE : tangent_at (z.A)
   L.TC   = C.xE : tangent_at (z.C)
   z.I    = intersection (L.TA,L.TC)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawCircles(w,E)
   \tkzDrawCircles[cyan](x,E)
   \tkzDrawCircles[red](G,A)
   \tkzDrawLines(A,I C,I F,I)
   \tkzDrawPoints(A,C,E,F)
   \tkzLabelPoints[right](A)
   \tkzLabelPoints[above right](E,F)
```

```
   \tkzLabelPoints[below](C)
\end{tikzpicture}
```



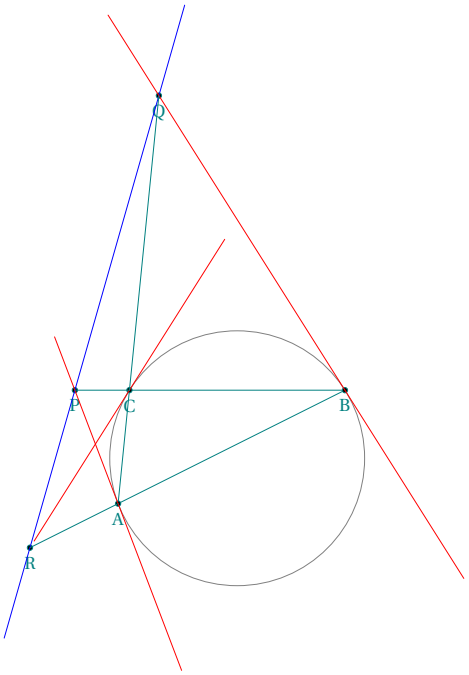
12.57 Midarc

```
\begin{tkzelements}
   z.A   = point:  new (-1,0)
   z.B   = point:  new (2,4)
   C.AB  = circle: new (z.A,z.B)
   z.C   =  z.A: rotation (math.pi/3,z.B)
   z.D   = C.AB: midarc (z.B,z.C)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPoints(A,B,C)
   \tkzDrawCircles(A,B)
   \tkzDrawPoints(A,...,D)
   \tkzLabelPoints(A,...,D)
\end{tikzpicture}
```



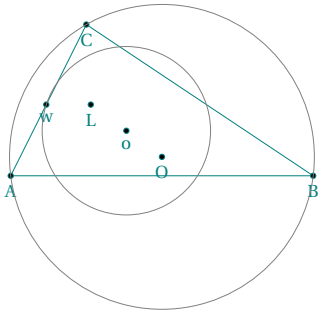
12.58 Lemoine Line without macro

```
\begin{tkzelements}
   scale       = 1.6
   z.A         = point: new (1,0)
   z.B         = point: new (5,2)
   z.C         = point: new (1.2,2)
   T           = triangle: new(z.A,z.B,z.C)
   z.O         = T.circumcenter
   L.AB        = line: new (z.A,z.B)
   L.AC        = line: new (z.A,z.C)
   L.BC        = line: new (z.B,z.C)
   C.OA        = circle: new (z.O,z.A)
   z.Ar,z.Al   = get_points (C.OA: tangent_at (z.A))
   z.Br,z.Bl   = get_points (C.OA: tangent_at (z.B))
   z.Cr,z.Cl   = get_points (C.OA: tangent_at (z.C))
   L.tA        = line: new (z.Ar,z.Al)
   L.tB        = line: new (z.Br,z.Bl)
   L.tC        = line: new (z.Cr,z.Cl)
   z.P         = intersection (L.tA,L.BC)
   z.Q         = intersection (L.tB,L.AC)
   z.R         = intersection (L.tC,L.AB)
\end{tkzelements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygon[teal](A,B,C)
   \tkzDrawCircle(O,A)
   \tkzDrawPoints(A,B,C,P,Q,R)
   \tkzLabelPoints(A,B,C,P,Q,R)
   \tkzDrawLine[blue](Q,R)
   \tkzDrawLines[red](Ar,Al Br,Q Cr,Cl)
   \tkzDrawSegments(A,R C,P C,Q)
\end{tikzpicture}
```



12.59 First Lemoine circle

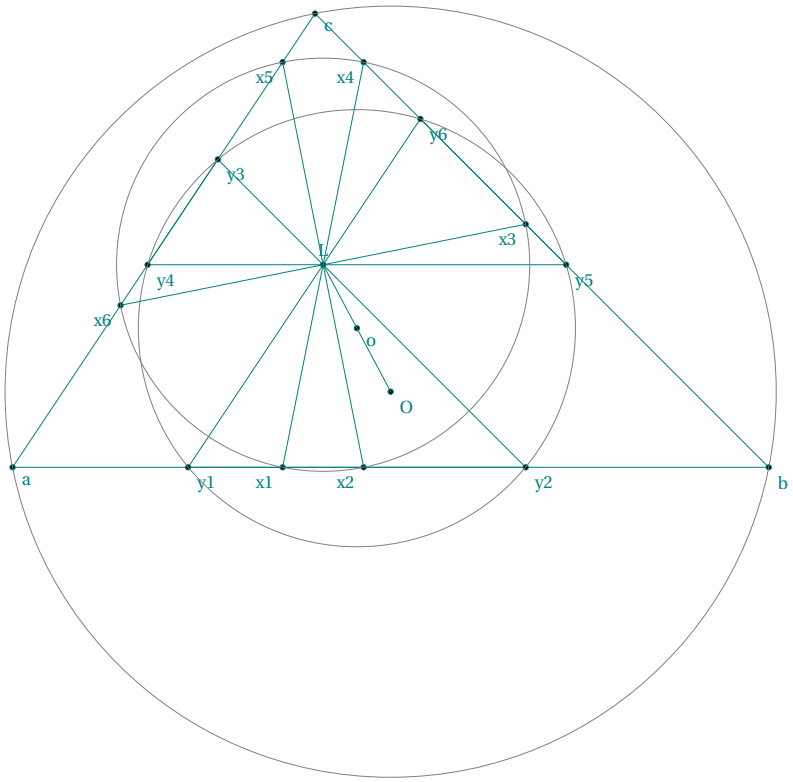
```
\begin{tkzelements}
   z.A      = point:  new (1,1)
   z.B      = point:  new (5,1)
   z.C      = point:  new (2,3)
   T        = triangle: new (z.A,z.B,z.C)
   z.O      = T.circumcenter
   z.o,z.w  = get_points (T : first_lemoine_circle ())
   z.L      = T : lemoine_point ()
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygons(A,B,C)
   \tkzDrawPoints(A,B,C,o,w,O,L)
   \tkzLabelPoints(A,B,C,o,w,O,L)
   \tkzDrawCircles(o,w  O,A)
\end{tikzpicture}
```



12.60 First and second Lemoine circles

```
\begin{tkzelements}
   scale             = 2
   z.a               = point:  new (0,0)
   z.b               = point:  new (5,0)
   z.c               = point:  new (2,3)
   T                 = triangle: new (z.a,z.b,z.c)
   z.O               = T.circumcenter
   z.o,z.p           = get_points (T : first_lemoine_circle ())
   L.ab              = line : new (z.a,z.b)
   L.ac              = line : new (z.a,z.c)
   L.bc              = line : new (z.b,z.c)
   z.L,z.x           = get_points (T : second_lemoine_circle ())
   C.first_lemoine   = circle : new (z.o,z.p)
   z.y1,z.y2         = intersection (L.ab,C.first_lemoine)
   z.y5,z.y6         = intersection (L.bc,C.first_lemoine)
   z.y3,z.y4         = intersection (L.ac,C.first_lemoine)
   C.second_lemoine  = circle : new (z.L,z.x)
   z.x1,z.x2         = intersection (L.ab,C.second_lemoine)
   z.x3,z.x4         = intersection (L.bc,C.second_lemoine)
   z.x5,z.x6         = intersection (L.ac,C.second_lemoine)
   L.y1y6            = line : new (z.y1,z.y6)
   L.y4y5            = line : new (z.y4,z.y5)
   L.y2y3            = line : new (z.y2,z.y3)
\end{tkzelements}
```

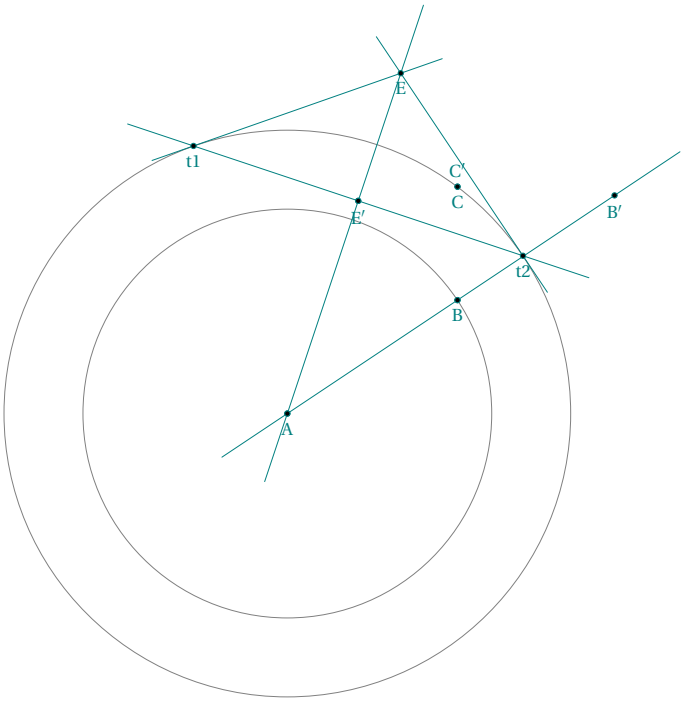
```
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygons(a,b,c y1,y2,y3,y4,y5,y6)
   \tkzDrawPoints(x1,x2,x3,x4,x5,x6,L)
   \tkzDrawPoints(a,b,c,o,O,y1,y2,y3,y4,y5,y6)
   \tkzLabelPoints[below right](a,b,c,o,O,y1,y2,y3,y4,y5,y6)
   \tkzLabelPoints[below left](x1,x2,x3,x4,x5,x6)
   \tkzLabelPoints[above](L)
   \tkzDrawCircles(L,x o,p O,a)
   \tkzDrawSegments(L,O x1,x4 x2,x5 x3,x6)
\end{tikzpicture}
```



12.61 Inversion

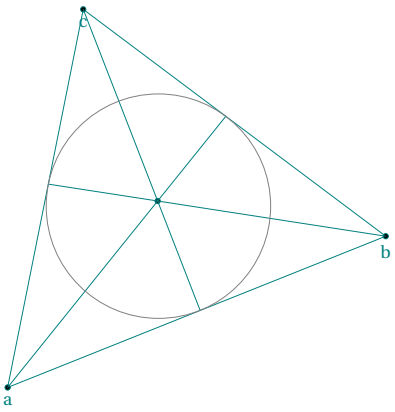
```
 \begin{tkzelements}
   z.A         = point: new (-1,0)
   z.B         = point: new (2,2)
   z.C         = point: new (2,4)
   z.E         = point: new (1,6)
   C.AC        = circle:    new (z.A,z.C)
   L.Tt1,L.Tt2 = C.AC: tangent_from (z.E)
   z.t1        = L.Tt1.pb
   z.t2        = L.Tt2.pb
   L.AE        = line: new (z.A,z.E)
   z.H         = L.AE : projection (z.t1)
   z.Bp,
   z.Ep,
   z.Cp        = C.AC: set_inversion ( z.B, z.E, z.C )
\end{tkzelements}
```

```
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPoints(A,B,C)
   \tkzDrawCircles(A,C A,B)
   \tkzDrawLines(A,B' E,t1 E,t2 t1,t2 A,E)
   \tkzDrawPoints(A,B,C,E,t1,t2,H,B',E')
   \tkzLabelPoints(A,B,C,E,t1,t2,B',E')
   \tkzLabelPoints[above](C')
\end{tikzpicture}
```



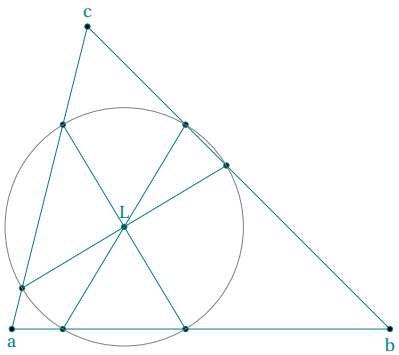
12.62 Gergonne point

```
\begin{tkzelements}
    z.a  = point: new(1,0)
    z.b  = point: new(6,2)
    z.c  = point: new(2,5)
    T    = triangle : new (z.a,z.b,z.c)
    z.g  = T : gergonne_point ()
    z.i  = T.incenter
    z.ta,z.tb,z.tc = get_points (T :  intouch ())
end{elements}
\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygons(a,b,c)
   \tkzDrawPoints(a,b,c,g)
   \tkzLabelPoints(a,b,c)
   \tkzDrawSegments (a,ta b,tb c,tc)
   \tkzDrawCircle(i,ta)
\end{tikzpicture}
```



12.63 Antiparallel through Lemoine point

```
\begin{tkzelements}
   z.a         = point:  new (0,0)
   z.b         = point:  new (5,0)
   z.c         = point:  new (1,4)
   T           = triangle: new (z.a,z.b,z.c)
   z.L         = T : lemoine_point ()
   L.anti      = T : antiparallel (z.L,0)
   z.x_0,z.x_1 = get_points (L.anti)
   L.anti      = T : antiparallel (z.L,1)
   z.y_0,z.y_1 = get_points (L.anti)
   L.anti      = T : antiparallel (z.L,2)
   z.z_0,z.z_1 = get_points (L.anti)
\end{tkzelements}

\begin{tikzpicture}
   \tkzGetNodes
   \tkzDrawPolygons(a,b,c)
   \tkzDrawPoints(a,b,c,L,x_0,x_1,y_0,y_1,z_0,z_1)
   \tkzLabelPoints(a,b)
   \tkzLabelPoints[above](L,c)
   \tkzDrawSegments(x_0,x_1 y_0,y_1 z_0,z_1)
   \tkzDrawCircle(L,x_0)
\end{tikzpicture}
```



Index