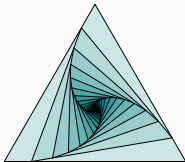


# tkz-elements

---



Alain Matthes

17 novembre 2023

AlterMundus

Questions via [al.ma@mac.com](mailto:al.ma@mac.com)

Exemples via [altermundus.fr](http://altermundus.fr)

# Plan de l'exposé

- ① Origine de **tkz-elements**
- ② Présentation
- ③ L'environnement **tkzelements**
- ④ La table **z**
- ⑤ Les objets
- ⑥ Les transferts
- ⑦ Exemples
- ⑧ Développement futur

# De TikZ vers tkz-elements

---



`veclen(x,y)` permet de calculer l'expression  $\sqrt{x^2 + y^2}$ .<sup>1</sup>

```
pgfmathparse{veclen(65,72)} \pgfmathresult
```

👉  $\sqrt{65^2 + 72^2} \approx 96,9884$  💣.

---

1. Ce calcul est obtenu à l'aide d'une approximation polynomiale, basée sur des idées de Rouben Rostamian.

On peut définir une macro `luaveclen` ainsi :

```
\def\luaveclen#1#2{\directlua{tex.print(string.format(
'\percentchar.5f',math.sqrt((#1)*(#1)+(#2)*(#2))))}}
```

👉  $\sqrt{65^2 + 72^2} = 97$  !!

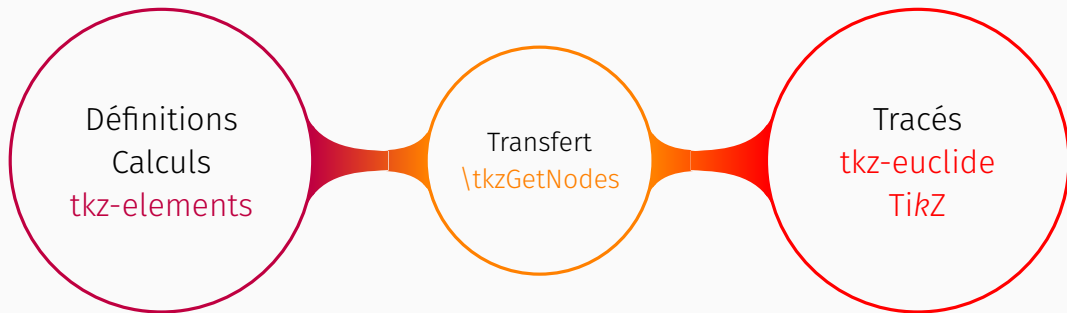
## Conclusion

- Difficile de faire de la géométrie sans la fonction **vec**len.
- La fonction **vec**len est très utilisée dans TikZ. Il faut prendre des précautions pour la redéfinir.
- Précision ou Efficacité ? (fp, xfp, fpu, autres ?)
- Solution : Lua $\text{\TeX}$  ? (usage local ? ou global ?)

# Présentation de tkz-elements

---





- Il faut compiler avec **Lua<sup>AT&T</sup>TeX** un fichier en **utf8**.
- Il faut charger **TikZ** ou **tkz-euclide**.
- Les calculs à l'aide de Lua se font dans un environnement **tkzelements**.
- Ensuite, dans un environnement **tikzpicture**, la macro **\tkzGetNodes** récupère les points définis dans l'environnement **tkzelements** pour en faire des **nodes**.

# Notions fondamentales

- Les définitions et les calculs sont effectués dans un repère orthonormal.
- Les fonctions reposent principalement sur les nombres complexes, les barycentres et les objets.
- Ces objets (voir section 5) sont pour le moment les suivants<sup>2</sup> : point, line, circle, triangle et ellipse.<sup>3</sup>

---

2. D'autres sont déjà en développement : matrices, vecteurs

3. Remerciements à **Nicolas Kisselhoff** pour les exemples de code Lua avec tkz-euclide, à **Roberto Giacomelli** pour le partage de ses documents sur l'utilisation des objets avec TikZ et enfin à **David Carlisle** pour ses réponses à mes multiples questions.

# Le cercle d'Apollonius

Cet exemple est celui qui est à l'origine de tkz-elements. Avec les premières versions, j'avais souvent des problèmes de précision (3). Cette version est celle qui utilise la plus simple méthode de construction et qui est possible grâce à Lua.

## Problème

Il s'agit de déterminer un cercle tangent intérieurement aux trois cercles exinscrits d'un triangle.

Remarque : Le cercle des neuf points ou cercle d'Euler est lui tangent extérieurement aux trois cercles. Les points de tangence forment le triangle de Feuerbach.

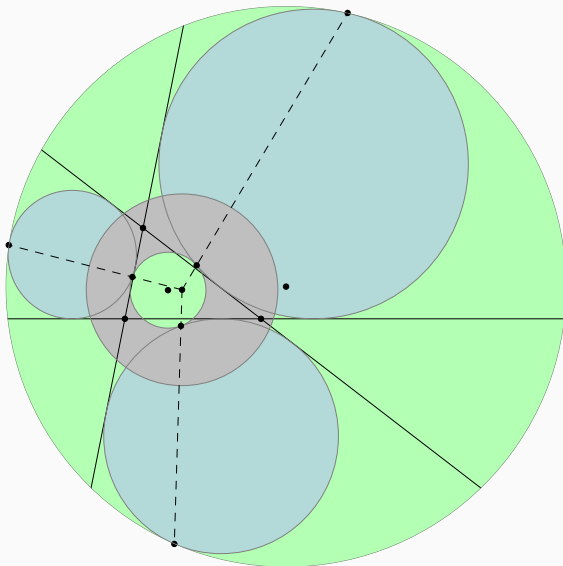
# Le cercle d'Apollonius

```
\begin{tkzelements}
  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,z.S      = T.ABC.eulercenter , T.ABC.spiekercenter
  z.Ea,z.Eb,z.Ec = get_points ( T.ABC : feuerbach () )
  z.Ja,z.Jb,z.Jc = get_points ( T.ABC : excentral () )
  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.through
  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}
```

# Le cercle d'Apollonius

```
\begin{tikzpicture}
  \tkzGetNodes
  \tkzFillCircles[green!30](O,xa)
  \tkzFillCircles[teal!30](Ja,Ea Jb,Eb Jc,Ec)
  \tkzFillCircles[lightgray](S,a)
  \tkzFillCircles[green!30](N,Ea)
  \tkzDrawPoints(xa,xb,xc)
  \tkzClipCircle(O,xa)
  \tkzDrawLines[add=3 and 3](A,B A,C B,C)
  \tkzDrawCircles(Ja,Ea Jb,Eb Jc,Ec S,a O,xa N,Ea)
  \tkzDrawPoints(O,A,B,C,S,Ea,Eb,Ec,N)
  \tkzDrawSegments[dashed](S,xa S,xb S,xc)
\end{tikzpicture}
```

# Le cercle d'Apollonius



# L'environnement tkzelements

---



# L'environnement tkzelements

L'environnement **tkzelements** encapsule un environnement **luacode**. Il fixe la valeur de **scale** à 1, puis il efface<sup>4</sup> les paires de chaque table.

L'environnement n'a actuellement pas d'option. Il est placé avant l'environnement **tikzpicture** ou bien à l'intérieur de celui-ci mais avant la macro `\tkzGetNodes`.

👉💣 Lorsque vous commentez des lignes de cet environnement, vous devez utiliser les deux tirets -- et non le symbole pourcent %, différence entre  $\text{\LaTeX}$  et Lua.

---

4. Voir la section Tables( 4) pour des explications sur cette ligne de codes

# La table z

---



# La table z

$z = \{\}$  crée une table vide.

```
z.A = point : new (1,2)
```

👉 L'écriture  $z.A$  est appelée **Sucre Syntaxique**. L'écriture principal étant 🧨  $z["A"]$ .

Ce code associe<sup>5</sup> au nom  $A$  (la clef : futur nom d'un node) un complexe  $1+2i$  (la valeur). Cela définit une paire  $\{A; 1+2i\}$  de la table  $z$ . La table nommée  $z$ <sup>6</sup> est constituée d'un ensemble non ordonné de paires. La table n'a pas de limite de taille.

Les autres objets (line, circle, triangle et ellipse) seront rangés dans des tables (L, C, T, E).

---

5. Une table est justement un tableau associatif qui lie une clef (une référence) à une valeur.

6.  $z$  en référence à la notation des affixes des nombres complexes.

# Les objets et leurs attributs

---



# Objet point

Lors de sa création, tous les attributs sont déterminés à l'aide de son affixe.

Création : `z.M = point : new (1,2)`<sup>7</sup>

Soit `z.M = 1+2i` si et seulement si `scale = 1`

| Attribut              | Commentaire                             | Exemple                                                                    |
|-----------------------|-----------------------------------------|----------------------------------------------------------------------------|
| <code>re, im</code>   | abscisse et coordonnée de A             | <code>z.M.re = 1</code> et <code>z.M.im = 2</code> <sup>8</sup>            |
| <code>type</code>     |                                         | <code>z.M.type = "point"</code>                                            |
| <code>argument</code> | $\alpha$ pente de $\overrightarrow{OM}$ | <code>z.M.argument = 0,46365 rad</code>                                    |
| <code>modulus</code>  | $OM = \sqrt{1^2 + 2^2}$                 | <code>z.M.modulus = <math>\sqrt{5} = 2,23607</math> cm</code> <sup>9</sup> |

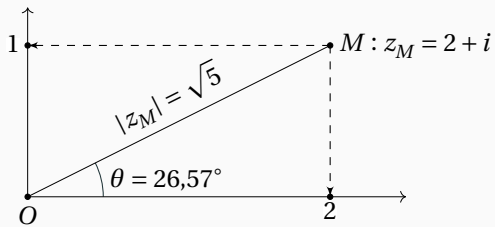
7. Possible `z.M = point: polar ( radius, phi )`

8. Partie réelle et partie imaginaire

9. `\directlua{tex.print(math.sqrt(5))}` donne 2,23607

# Objet point

`z.M = point: new (2,1)`



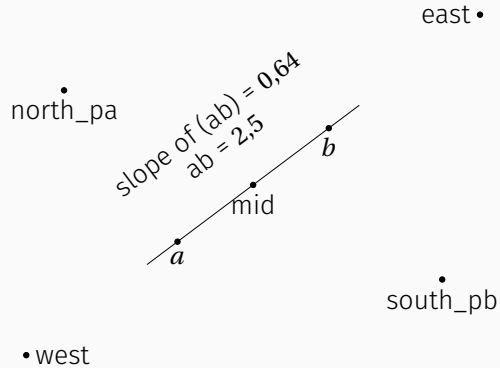
Création : `L.AB = line : new (z.A,z.B);` droite passant par les points A et B.<sup>10</sup>

| Attribut              | Commentaire                | Exemple                                 |
|-----------------------|----------------------------|-----------------------------------------|
| <code>pa</code>       | premier point de la droite | <code>L.AB.pa = z.A</code>              |
| <code>pb</code>       | dernier point de la droite | <code>L.AB.pb = z.B</code>              |
| <code>type</code>     |                            | <code>z.A.type = "line"</code>          |
| <code>mid</code>      | milieu de [AB]             | <code>z.M = L.AB.mid</code>             |
| <code>north_pa</code> |                            | voir <sup>11</sup> voir exemple suivant |
| <code>slope</code>    | pente de (AB)              | <code>L.AB.slope</code>                 |
| <code>length</code>   | AB                         | <code>AB = L.AB.length</code>           |

10. Cela définit aussi le segment [AB]

11. Et aussi `south_pa`, `north_pb`, `south_pb`, `west`, `east`

# Attributs de l'objet line



# Objet circle

Création : `C.OA = circle : new (z.O,z.A)`<sup>12</sup>; centre *O*, passant par *A*.<sup>13</sup>

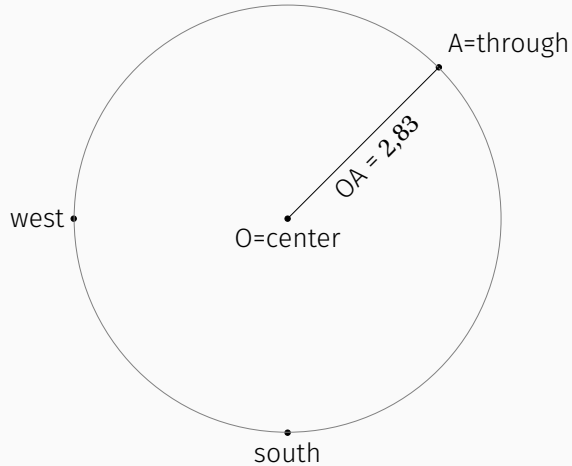
| Attribut             | Commentaire      | Exemple                                          |
|----------------------|------------------|--------------------------------------------------|
| <code>center</code>  | centre du cercle | <code>z.O = C.OA.center</code>                   |
| <code>through</code> | point du cercle  | <code>z.A = C.OA.through</code>                  |
| <code>type</code>    |                  | <code>C.OA.type = "circle"</code>                |
| <code>radius</code>  | rayon            | <code>OA = C.OA.radius</code> OA est nombre réel |
| <code>south</code>   | point cardinal   | <code>z.S = C.OA.south</code> <sup>14</sup>      |

12. Possible `C.OA = circle : radius (z.O, r)`

13. Je privilégie cette méthode car elle correspond aux constructions avec le compas.

14. Également `north, east, west`

## Attributs de l'objet circle



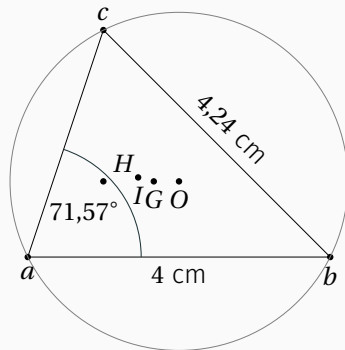
# Objet triangle

Création : `T.EFG = line : new (z.E,z.F,z.G).`

| Attribut                        | Commentaire                      | Exemple                               |
|---------------------------------|----------------------------------|---------------------------------------|
| <code>pa,pb,pc</code>           | les sommets <sup>15</sup>        | <code>z.F = T.EFG.pb</code>           |
| <code>type</code>               |                                  | <code>T.EFG.type = "triangle"</code>  |
| <code>circumcenter</code>       | centre cercle circonscrit        | <code>T.EFG.circumcenter</code>       |
| <code>centroid</code>           | centre de gravité                |                                       |
| <code>incenter</code>           | centre du cercle inscrit         |                                       |
| <code>orthocenter</code>        | intersection of altitudes        |                                       |
| <code>eulercenter</code>        | centre du cercle des neuf points |                                       |
| <code>a, b, c</code>            | longueur des côtés               | <code>T.EFG.a</code>                  |
| <code>alpha, beta, gamma</code> | angles aux sommets               | Angle en E = <code>T.EFG.alpha</code> |
| <code>ab, ac, bc</code>         | droites <u>par</u> les sommets   | <code>L.EG = T.EFG.ac.mid</code>      |

15. Dans l'ordre de la création.

# Attributs de l'objet triangle



# Objet ellipse

Création : `E.CAB = ellipse : new (z.C,z.A,z.B)`<sup>16</sup>; Ellipse dont le centre est *C*.  
*A* et *B* sont des extrémités du grand et du petit axe.

| Attribut                                       | Commentaire                | Exemple                                                                |
|------------------------------------------------|----------------------------|------------------------------------------------------------------------|
| <code>center, vertex, ...</code> <sup>17</sup> | centre, vertex, covertex   | <code>E.CAB.center = z.C</code><br><code>E.CAB.type = "ellipse"</code> |
| <code>Rx, Ry</code>                            | rayons grand et petit axes | <code>CA = E.CAB.Rx</code>                                             |
| <code>slope</code>                             | pente du grand axe         | <code>E.CAB.slope = pente de <math>(\overrightarrow{CA})</math></code> |
| <code>Fa, Fb</code>                            | les foyers                 |                                                                        |
| <code>north</code>                             | points cardinaux           | <sup>18</sup>                                                          |

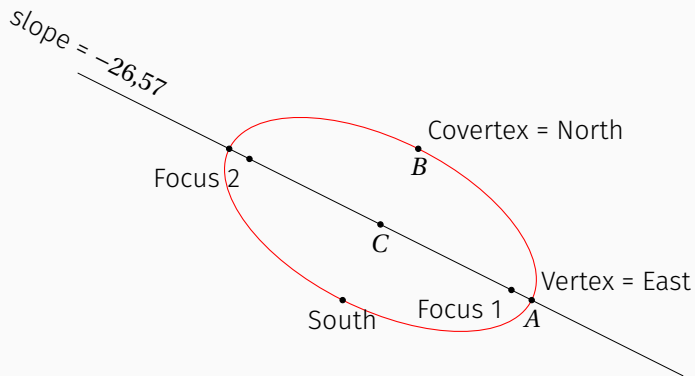
16. Deux autres méthodes existent `E.C = ellipse : foci (Fa,Fb,V)` (Deux foyers et le vertex) ou bien

`E.C = ellipse : radii (c,a,b,sl)` (le centre, deux rayons et la pente du grand axe.)

17. vertex et covertex sont les extrémités des axes.

18. Et aussi `south, east, west`.

## Attributs de l'objet ellipse



# Objets et méthodes

---



Une méthode<sup>19</sup> est une fonction liée à un objet. Nous avons vu précédemment

```
C.OH = T.ABC : in_circle ()  
L.MI = L.AB : ortho_from (z.M)
```

Les méthodes peuvent retourner toutes sortes de valeurs. Elles utilisent parfois des paramètres. Avec **tkz-elements**, elles retournent, soit des objets parmi ceux cités, soit des valeurs numériques, soit des booléens.

---

19. Voir la documentation du package pour la description de toutes les méthodes.

# Transferts

---



## \tkzGetnodes

La table `z` est parcourue. Les paires (Clef/Valeur) sont extraites de la table, une à une.  $K$ <sup>20</sup> est attribuée au nom du node<sup>21</sup>, `V.re` et `V.im` sont les coordonnées de celui-ci.<sup>22</sup>

```
\def\tkzGetNodes{\directlua{%  
  for K,V in pairs(z) do  
    tex.print("\coordinate ("..K..) at ("..V.re..","..V.im..");  
    \\\")  
  end}}
```

---

20. Si la clef `K` se termine par la lettre `p` (minuscule) alors la lettre `p` sera remplacée par un prime `'`

21. `\pgfnodealias{new name}{existing node}` est une possibilité pour modifier le nom d'un node

22. Rappel : `V` est l'afixe du point.

Il est possible d'extraire des valeurs en utilisant la macro `\tkzUseLua` :

```
\def\tkzUseLua#1{\directlua{tex.print(tostring(#1))}}
```

```
\begin{tkzelements}
  z.A      = point : new (1,3)
  z.B      = point : new (4,-2)
  z.C      = point : new (2,4)
  L.AB     = line  : new (z.A,z.B)
  T.ABC    = triangle : new (z.A,z.B,z.C)
\end{tkzelements}
```

```
\tkzUseLua{L.AB.length} donne 5,83 cm
\tkzUseLua{T.ABC.alpha}}  donne 1,82 rad
```

# Examples

---



# Division harmonique

```
\begin{tkzelements}
  z.A    = point : new(0,0)
  z.B    = point : new(5,0)
  L.AB   = line  : new (z.A,z.B)
  z.C    = L.AB : gold_ratio ()
\end{tkzelements}
```



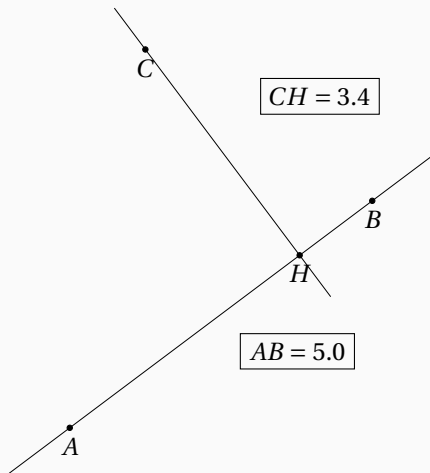
$$\frac{AC+CB}{AC} = \frac{AC}{CB} = \varphi = \frac{1+\sqrt{5}}{2} \approx 1,61803$$

Pour le calcul de  $\varphi$  🐞 : `\directlua{tex.print((1+math.sqrt(5))/2)}`.

## Distance d'un point à une droite

```
\begin{tkzelements}
  z.A      = point : new (0 , 0)
  z.B      = point : new (4 , 3)
  z.C      = point : new (1 , 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}
et
\tkzLabelSegment[above right=2em,draw](C,H){$CH = \tkzUseLua{d}$}
```

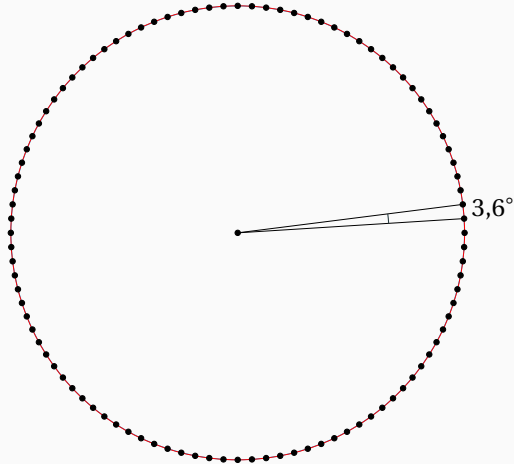
## Distance d'un point à une droite



## Boucle et notation

```
\begin{tkzelements}
  local r    = 3
  z.0        = point : new (0,0)
  max        = 100
  for i      = 1,max
  do
    z["A_"..i] = point : polar(r,2*i*math.pi/max)
  end
  a = math.deg(get_angle (z.0,z.A_1,z.A_2))
\end{tkzelements}
```

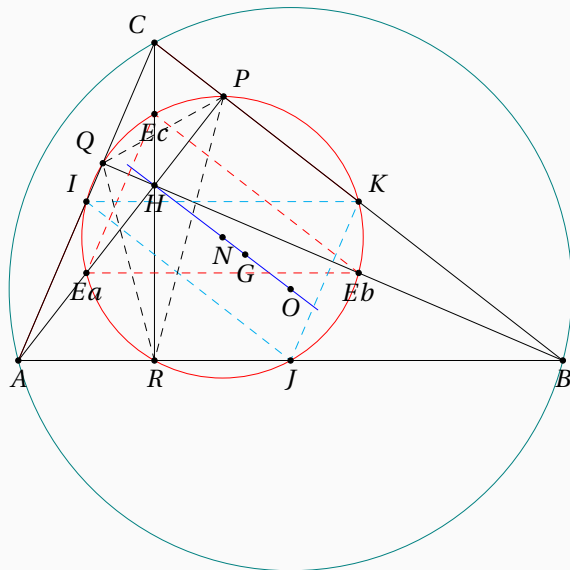
# Boucle et notation d'une table



## Cercle d'Euler ou des « neuf points »

```
\begin{tkzelements}
  scale          = 1.2
  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 ())
  z.Ea,z.Eb,z.Ec = get_points (T.ABC : euler ())
\end{tkzelements}
```

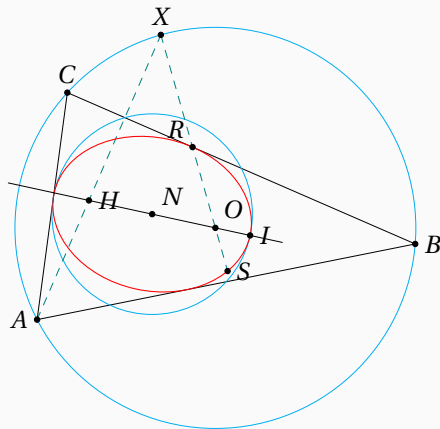
## Cercle d'Euler ou « neuf points »



## Ellipse dans un triangle

```
\begin{tkzelements}
  z.A,z.B      = point: new (0 , 0),point: new (10 , 2)
  z.C          = point: new (.8 , 6)
  T.ABC        = triangle: new (z.A,z.B,z.C)
  L.euler      = T.ABC: euler_line ()
  z.H,z.O      = L.euler.pa,L.euler.pb
  z.N          = T.ABC.eulercenter
  C.circum     = circle: new (z.O,z.A)
  C.euler      = circle: new (z.N,T.ABC: medial () .pc )
  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)
  z.X          = intersection (line: new (z.A,z.H),C.circum)
  z.R,z.S      = intersection (line: new (z.X,z.O),E)
  a,b,ang      = E.Rx,E.Ry,math.deg(E.slope)
\end{tkzelements}
```

# Ellipse dans un triangle

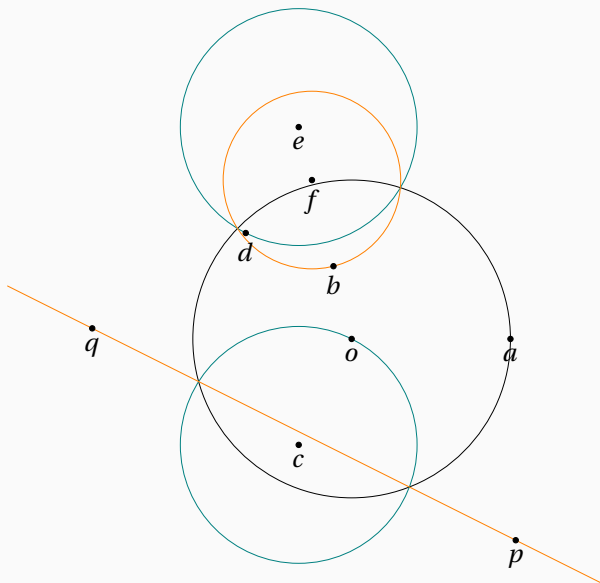


Ce code peut conduire à la création de l'hexagramme de **Pascal**.

## Inversion par rapport à un cercle

```
\begin{tkzelements}
z.o,z.a  = point:  new (-1,3),point:  new (2,3)
z.c      = point:  new (-2,1)
z.e,z.d  = point:  new (-2,7),point:  new (-3,5)
C.oa     = circle: new (z.o,z.a)
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.O,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.O,z.H = get_points(obj) end
color = "orange"
\end{tkzelements}
```

# Inversion par rapport à un cercle



# Développement futur

---



# Développement futur

- Introduction des matrices
- Résolution d'équations
- Parabole, hyperbole
- Vecteurs

FIN

---



Questions?