tkz-euclide 4.00

AlterMundus

tkz-euclide

tool for

Euclidean Geometry

Alain Matthes

January 4, 2022 Documentation V.4.00

http://altermundus.fr

tkz-euclide

AlterMundus

Alain Matthes

☞ `tkz-euclide` 4.00 is now independent of tkz-base. It is a set of convenient macros for drawing in a plane (fundamental two-dimensional object) with a Cartesian coordinate system. It handles the most classic situations in Euclidean Geometry. `tkz-euclide` is built on top of PGF and its associated front-end Ti*k*Z and is a (La)TeX-friendly drawing package. The aim is to provide a high-level user interface to build graphics relatively simply. The idea is to allow you to follow step by step a construction that would be done by hand as naturally as possible.
English is not my native language so there might be some errors.

☞ Firstly, I would like to thank **Till Tantau** for the beautiful LaTeX package, namely Ti*k*Z.

☞ Acknowledgements : I received much valuable advice, remarks, corrections and examples from **Jean-Côme Charpentier**, **Josselin Noirel**, **Manuel Pégourié-Gonnard**, **Franck Pastor**, **David Arnold**, **Ulrike Fischer**, **Stefan Kottwitz**, **Christian Tellechea**, **Nicolas Kisselhoff**, **David Arnold**, **Wolfgang Büchel**, **John Kitzmiller**, **Dimitri Kapetas**, **Gaétan Marris**, **Mark Wibrow**, **Yves Combe** for his work on a protractor, **Paul Gaborit**, **Laurent Van Deik** for all his corrections, remarks and questions and **Muzimuzhi Z** for the code about the option "dim".

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

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

Please report typos or any other comments to this documentation to: Alain Matthes.
This file can be redistributed and/or modified under the terms of the LaTeX Project Public License Distributed from CTAN archives.

Contents

I. General survey : a brief but comprehensive review 13

1. Installation 14

2. Presentation and Overview 15
- 2.1. Why tkz-euclide? . . . 15
- 2.2. TikZ vs tkz-euclide . . . 15
  - 2.2.1. Book I, proposition I _Euclid's Elements_ . . . 15
  - 2.2.2. Complete code with tkz-euclide . . . 17
- 2.3. tkz-euclide4 vs tkz-euclide3 . . . 19
- 2.4. How to use the tkz-euclide package ? . . . 20
  - 2.4.1. Let's look at a classic example . . . 20
  - 2.4.2. Part I: golden triangle . . . 21
  - 2.4.3. Part II: two others methods with golden and euclid triangle . . . 23
  - 2.4.4. Complete but minimal example . . . 24

3. The Elements of tkz code 27
- 3.1. Objects and language . . . 27
- 3.2. Notations and conventions . . . 27
- 3.3. Set, Calculate, Draw, Mark, Label . . . 29

4. News and compatibility 30

II. Setting 32

5. First step: fixed points 33

6. Definition of a point : \tkzDefPoint or \tkzDefPoints 33
- 6.1. Defining a named point \tkzDefPoint . . . 34
  - 6.1.1. Cartesian coordinates . . . 34
  - 6.1.2. Calculations with xfp . . . 35
  - 6.1.3. Polar coordinates . . . 35
  - 6.1.4. Relative points . . . 35
- 6.2. Point relative to another: \tkzDefShiftPoint . . . 35
  - 6.2.1. Isosceles triangle . . . 36
  - 6.2.2. Equilateral triangle . . . 36
  - 6.2.3. Parallelogram . . . 36
- 6.3. Definition of multiple points: \tkzDefPoints . . . 37
- 6.4. Create a triangle . . . 37
- 6.5. Create a square . . . 37

III. Calculating 38

7. Auxiliary tools 39
- 7.1. Constants . . . 39
- 7.2. New point by calculation . . . 39

8. Special points 40
- 8.1. Middle of a segment \tkzDefMidPoint . . . 40
  - 8.1.1. Use of \tkzDefMidPoint . . . 40

8.2. Barycentric coordinates . . . 41
8.2.1. Using `\tkzDefBarycentricPoint` with two points . . . 41
8.2.2. Using `\tkzDefBarycentricPoint` with three points . . . 41
8.3. Golden ratio . . . 42
8.4. Use the golden ratio to divide a line segment . . . 42
8.5. Internal Similitude Center . . . 42
8.6. Equidistant points . . . 43
8.6.1. `\tkzDefEquiPoints` . . . 43
8.6.2. Using `\tkzDefEquiPoints` with options . . . 43

9. Special points relating to a triangle 44
9.1. Triangle center: `\tkzDefTriangleCenter` . . . 44
9.1.1. Option **ortho** or **orthic** . . . 44
9.1.2. Option **centroid** . . . 45
9.1.3. Option **circum** . . . 45
9.1.4. Option **in** . . . 45
9.1.5. Option **ex** . . . 45
9.1.6. Option **euler** . . . 46
9.1.7. Option **symmedian** . . . 46
9.1.8. Option **spieker** . . . 47
9.1.9. Option **gergonne** . . . 47
9.1.10. Option **nagel** . . . 48
9.1.11. Option **mittenpunkt** . . . 48
9.1.12. Example : relation between **gergonne**, **centroid** and **mittenpunkt** . . . 49

10. Projection of excenters 50
10.0.1. Excircles . . . 50

11. Point on line or circle 51
11.1. Point on a line . . . 51
11.1.1. Use of option **pos** . . . 51
11.2. Point on a circle . . . 52

12. Definition of points by transformation : `\tkzDefPointBy` 53
12.1. Examples of transformations . . . 53
12.1.1. translation . . . 53
12.1.2. reflection (orthogonal symmetry) . . . 54
12.1.3. **homothety** and **projection** . . . 54
12.1.4. **projection** . . . 54
12.1.5. **symmetry** . . . 55
12.1.6. **rotation** . . . 55
12.1.7. **rotation in radian** . . . 55
12.1.8. **inversion** . . . 55
12.1.9. Inversion of lines . . . 57
12.1.10. Inversion of circle . . . 58
12.1.11. Inversion of Triangle with respect to the Incircle . . . 58
12.1.12. Inversion: orthogonal circle with inversion circle . . . 58
12.1.13. **Inversion** and **homothety** . . . 59
12.1.14. **inversion negative** . . . 59
12.2. Transformation of multiple points; `\tkzDefPointsBy` . . . 60
12.2.1. Example of translation . . . 61
12.2.2. Example of symmetry . . . 61

13. Defining points using a vector 62
13.1. \tkzDefPointWith . . . 62
13.1.1. Option **colinear at**, simple example . . . 62
13.1.2. Option **colinear at**, complex example . . . 63
13.1.3. Option **colinear at** . . . 63
13.1.4. Option **colinear at** . . . 64
13.1.5. Option **orthogonal** . . . 64
13.1.6. Option **orthogonal** . . . 64
13.1.7. Option **orthogonal** more complicated example . . . 65
13.1.8. Options **colinear** and **orthogonal** . . . 65
13.1.9. Option **orthogonal normed** . . . 65
13.1.10. Option **orthogonal normed** and K=2 . . . 65
13.1.11. Option **linear** . . . 66
13.1.12. Option **linear normed** . . . 66
13.2. \tkzGetVectxy . . . 66
13.2.1. Coordinate transfer with \tkzGetVectxy . . . 67

14. The straight lines 68
14.1. Definition of straight lines . . . 68
14.1.1. Example with **mediator** . . . 68
14.1.2. Example with **bisector** and **normed** . . . 69
14.1.3. Example with **orthogonal** and **parallel** . . . 69
14.1.4. An envelope . . . 69
14.1.5. A parabola . . . 70
14.2. Specific lines: Tangent to a circle . . . 70
14.2.1. Example of a tangent passing through a point on the circle . . . 71
14.2.2. Example of tangents passing through an external point . . . 71
14.2.3. Example of Andrew Mertz . . . 72
14.2.4. Drawing a tangent option **from** . . . 72

15. Triangles 73
15.1. Definition of triangles \tkzDefTriangle . . . 73
15.1.1. Option **two angles** . . . 74
15.1.2. Option **school** . . . 74
15.1.3. Option **pythagore** . . . 74
15.1.4. Option **golden** . . . 75
15.1.5. Option **equilateral** and **isosceles right** . . . 75
15.1.6. Option **gold** . . . 75
15.1.7. Option **euclid** . . . 75

16. Specific triangles with \tkzDefSpcTriangle 76
16.1. How to name the vertices . . . 76
16.2. Option **medial** or **centroid** . . . 76
16.3. Option **in** or **incentral** . . . 77
16.4. Option **ex** or **excentral** . . . 77
16.5. Option **intouch** or **contact** . . . 78
16.6. Option **extouch** . . . 78
16.7. Option **orthic** . . . 79
16.8. Option **feuerbach** . . . 80
16.9. Option **tangential** . . . 80
16.10. Option **euler** . . . 80
16.11. Option **euler** and Option **orthic** . . . 82
16.12. Option **symmedial** . . . 83

17. Definition of polygons 84
17.1. Defining the points of a square . . . 84
17.1.1. Using \tkzDefSquare with two points . . . 84
17.1.2. Use of \tkzDefSquare to obtain an isosceles right-angled triangle . . . 84
17.1.3. Pythagorean Theorem and \tkzDefSquare . . . 85
17.2. Defining the points of a rectangle . . . 85
17.2.1. Example of a rectangle definition . . . 85
17.3. Definition of parallelogram . . . 85
17.3.1. Example of a parallelogram definition . . . 86
17.4. The golden rectangle . . . 86
17.4.1. Golden Rectangles . . . 86
17.4.2. Construction of the golden rectangle . . . 86
17.5. Regular polygon . . . 87
17.5.1. Option **center** . . . 87
17.5.2. Option **side** . . . 88

18. The Circles 89
18.1. Characteristics of a circle: \tkzDefCircle . . . 89
18.1.1. Example with a random point and option **through** . . . 90
18.1.2. Example with option **diameter** . . . 90
18.1.3. Circles inscribed and circumscribed for a given triangle . . . 90
18.1.4. Example with option **ex** . . . 90
18.1.5. Euler's circle for a given triangle with option **euler** . . . 91
18.1.6. Apollonius circles for a given segment option **apollonius** . . . 92
18.1.7. Circles exinscribed to a given triangle option **ex** . . . 92
18.1.8. Spieker circle with option **spieker** . . . 92
18.1.9. Examples from js bibra tex.stackexchange.com . . . 93

19. Definition of circle by transformation; \tkzDefCircleBy 95
19.1. Examples of transformations . . . 96
19.1.1. **Translation** . . . 96
19.1.2. **Reflection** (orthogonal symmetry) . . . 96
19.1.3. **Homothety** . . . 96
19.1.4. **Symmetry** . . . 97
19.1.5. **Rotation** . . . 97
19.1.6. **Orthogonal from** . . . 97
19.1.7. **Orthogonal from** : Right angle between circles . . . 97
19.1.8. **Orthogonal through** . . . 98
19.1.9. **Inversion** . . . 98

20. Intersections 99
20.1. Intersection of two straight lines . . . 99
20.1.1. Example of intersection between two straight lines . . . 99
20.2. Intersection of a straight line and a circle . . . 99
20.2.1. Line-circle intersection . . . 100
20.2.2. Line-circle intersection in Sangaku . . . 100
20.2.3. More complex example of a line-circle intersection . . . 101
20.2.4. Circle defined by a center and a measure, and special cases . . . 102
20.2.5. More complex example . . . 102
20.2.6. Calculation of radius example 1 . . . 103
20.2.7. Calculation of radius example 2 . . . 103
20.2.8. Option "with nodes" . . . 104

20.3. Intersection of two circles 104
20.3.1. Construction of an equilateral triangle 104
20.3.2. Example a mediator 105
20.3.3. An isosceles triangle. 105
20.3.4. Segment trisection 105
20.3.5. With the option "**with nodes**" 106
20.3.6. Mix of intersections 107
20.3.7. An oval 107

21. The angles 108
21.1. Recovering an angle `\tkzGetAngle` 108
21.2. Example of the use of `\tkzGetAngle` 108
21.3. Angle formed by three points 108
21.3.1. Verification of angle measurement 109
21.3.2. Determination of the three angles of a triangle 109
21.3.3. Angle between two circles 110
21.4. Angle formed by a straight line with the horizontal axis `\tkzFindSlopeAngle` 110
21.4.1. Use of `\tkzFindSlopeAngle` and `\tkzGetAngle` 110
21.4.2. Another use of `\tkzFindSlopeAngle` 111

22. Random point definition 112
22.1. Obtaining random points 112
22.2. Random point in a rectangle 112
22.3. Random point on a segment 113
22.4. Random point on a straight line 113
22.4.1. Random point on a circle 113
22.4.2. Random example and circle of Apollonius 114
22.5. Middle of a compass segment 114

IV. Drawing and Filling 116

23. Drawing 117
23.1. Draw a point or some points 117
23.1.1. Drawing points `\tkzDrawPoint` 117
23.1.2. Example of point drawings 117
23.1.3. Example 118

24. Drawing the lines 118
24.1. Draw a straight line 118
24.1.1. Examples with `add` 119
24.1.2. Example with `\tkzDrawLines` 119
24.1.3. Example with the option `add` 120

25. Drawing a segment 120
25.1. Draw a segment `\tkzDrawSegment` 120
25.1.1. Example with point references 120
25.1.2. Example of extending an segment with option `add` 121
25.1.3. Adding dimensions with option `dim` partI 121
25.1.4. Adding dimensions with option `dim` part II 122
25.2. Drawing segments `\tkzDrawSegments` 122
25.2.1. Place an arrow on segment 122
25.3. Drawing line segment of a triangle 123
25.3.1. How to draw `Altitude` 123

25.4. Drawing a polygon . . . 123
25.4.1. \tkzDrawPolygon . . . 123
25.4.2. Option **two angles** . . . 124
25.4.3. Style of line . . . 124
25.5. Drawing a polygonal chain . . . 124
25.5.1. Polygonal chain . . . 125
25.5.2. The idea is to inscribe two squares in a semi-circle. . . . 125
25.5.3. Polygonal chain: index notation . . . 125

26. Draw a circle with \tkzDrawCircle 125
26.1. Draw one circle . . . 125
26.1.1. Circles and styles, draw a circle and color the disc . . . 126
26.2. Drawing circles . . . 126
26.2.1. Circles defined by a triangle. . . . 127
26.2.2. Concentric circles. . . . 127
26.2.3. Exinscribed circles. . . . 128
26.2.4. Cardioid . . . 128
26.3. Drawing semicircle . . . 129
26.3.1. Use of \tkzDrawSemiCircle . . . 129
26.4. Drawing semicircles . . . 129

27. Drawing arcs 129
27.1. Option **towards** . . . 130
27.2. Option **towards** . . . 130
27.3. Option **rotate** . . . 131
27.4. Option **R** . . . 131
27.5. Option **R with nodes** . . . 131
27.6. Option **delta** . . . 131
27.7. Option **angles**: example 1 . . . 132
27.8. Option **angles**: example 2 . . . 133

28. Drawing a sector or sectors 133
28.1. \tkzDrawSector . . . 133
28.1.1. \tkzDrawSector and **towards** . . . 133
28.1.2. \tkzDrawSector and **rotate** . . . 134
28.1.3. \tkzDrawSector and **R** . . . 134
28.1.4. \tkzDrawSector and **R** . . . 135
28.1.5. \tkzDrawSector and **R with nodes** . . . 135
28.2. Coloring a disc . . . 136
28.2.1. Yin and Yang . . . 136
28.2.2. From a sangaku . . . 136
28.2.3. Clipping and filling part I . . . 137
28.2.4. Clipping and filling part II . . . 137
28.2.5. Clipping and filling part III . . . 138
28.3. Coloring a polygon . . . 138
28.3.1. \tkzFillPolygon . . . 138
28.4. \tkzFillSector . . . 138
28.4.1. \tkzFillSector and **towards** . . . 139
28.4.2. \tkzFillSector and **rotate** . . . 139
28.5. Colour an angle: \tkzFillAngle . . . 139
28.5.1. Example with **size** . . . 140
28.5.2. Changing the order of items . . . 140
28.5.3. Multiples angles . . . 141

29. Controlling Bounding Box 142
29.1. Utility of \tkzInit . . . 142
29.2. \tkzInit . . . 142
29.3. \tkzClip . . . 142
29.4. \tkzClip and the option space . . . 143
29.5. tkzShowBB . . . 143
29.5.1. Example with \tkzShowBB . . . 144
29.6. tkzClipBB . . . 144
29.6.1. Example with \tkzClipBB and the bisectors . . . 144

30. Clipping different objects 145
30.1. Clipping a polygon . . . 145
30.1.1. \tkzClipPolygon . . . 145
30.1.2. \tkzClipPolygon[out] . . . 145
30.1.3. Example: use of ”Clip” for Sangaku in a square . . . 146
30.2. Clipping a disc . . . 146
30.2.1. Simple clip . . . 146
30.3. Clip out . . . 147
30.4. Intersection of disks . . . 147
30.5. Clipping a sector . . . 147
30.5.1. Example 1 . . . 148
30.5.2. Example 2 . . . 148
30.6. Options from TikZ: trim left or right . . . 148
30.7. TikZ Controls \pgfinterruptboundingbox and \endpgfinterruptboundingbox . . . 148
30.7.1. Example about contolling the bouding box . . . 149
30.8. Reverse clip: tkzreverseclip . . . 149
30.8.1. Example with \tkzClipPolygon[out] . . . 149

V. Marking 151
30.9. Mark a segment \tkzMarkSegment . . . 152
30.9.1. Several marks . . . 152
30.9.2. Use of mark . . . 152
30.10. Marking segments \tkzMarkSegments . . . 152
30.10.1. Marks for an isosceles triangle . . . 152
30.11. Another marking . . . 153
30.12. Mark an arc \tkzMarkArc . . . 153
30.12.1. Several marks . . . 153
30.13. Mark an angle mark : \tkzMarkAngle . . . 153
30.13.1. Example with mark = x . . . 154
30.13.2. Example with mark = ... 154
30.14. Marking a right angle: \tkzMarkRightAngle . . . 154
30.14.1. Example of marking a right angle . . . 155
30.14.2. Example of marking a right angle, german style . . . 155
30.14.3. Mix of styles . . . 155
30.14.4. Full example . . . 156
30.15. \tkzMarkRightAngles . . . 156

VI. Labelling 157

31. Labelling 158
31.1. Label for a point . . . 158
31.1.1. Example with \tkzLabelPoint . . . 158
31.1.2. Label and reference . . . 158
31.2. Add labels to points \tkzLabelPoints . . . 158
31.2.1. Example with \tkzLabelPoints . . . 159
31.3. Automatic position of labels \tkzAutoLabelPoints . . . 159
31.3.1. Example with \tkzAutoLabelPoints . . . 159
31.3.2. Example with \tkzAutoLabelPoints . . . 159

32. Label for a segment 160
32.0.1. First example . . . 160
32.0.2. Example : blackboard . . . 161
32.0.3. Labels and option : swap . . . 161
32.0.4. Labels for an isosceles triangle . . . 162

33. Add labels on a straight line \tkzLabelLine 162
33.0.1. Example with \tkzLabelLine . . . 162
33.1. Label at an angle : \tkzLabelAngle . . . 162
33.1.1. Example author js bibra stackexchange . . . 163
33.1.2. Example with pos . . . 163
33.2. Giving a label to a circle . . . 164
33.2.1. Example . . . 165
33.2.2. Second example . . . 165

34. Label for an arc 165
34.0.1. Label on arc . . . 166

VII. Complements 167

35. Using the compass 168
35.1. Main macro \tkzCompass . . . 168
35.1.1. Option length . . . 168
35.1.2. Option delta . . . 168
35.2. Multiple constructions \tkzCompasss . . . 168

36. The Show 170
36.1. Show the constructions of some lines \tkzShowLine . . . 170
36.1.1. Example of \tkzShowLine and parallel . . . 170
36.1.2. Example of \tkzShowLine and perpendicular . . . 170
36.1.3. Example of \tkzShowLine and bisector . . . 171
36.1.4. Example of \tkzShowLine and mediator . . . 171
36.2. Constructions of certain transformations \tkzShowTransformation . . . 171
36.2.1. Example of the use of \tkzShowTransformation . . . 172
36.2.2. Another example of the use of \tkzShowTransformation . . . 172

37. Protractor 173
37.1. The circular protractor . . . 173
37.2. The circular protractor, transparent and returned . . . 173

38. Miscellaneous tools 174
38.1. Duplicate a segment . . . 174
38.1.1. Proportion of gold with \tkzDuplicateSegment . . . 174
38.1.2. Golden triangle or sublime triangle . . . 175
38.2. Segment length \tkzCalcLength . . . 175
38.2.1. Compass square construction . . . 176
38.2.2. Example . . . 176
38.3. Transformation from pt to cm or cm to pt . . . 176
38.4. Change of unit . . . 177
38.5. Get point coordinates . . . 177
38.5.1. Coordinate transfer with \tkzGetPointCoord . . . 177
38.5.2. Sum of vectors with \tkzGetPointCoord . . . 177

VIII. Working with style 178

39. Predefined styles 179

40. Points style 179
40.0.1. Use of \tkzSetUpPoint . . . 179
40.0.2. Global style or local style . . . 180
40.0.3. Simple example with \tkzSetUpPoint . . . 180
40.0.4. Use of \tkzSetUpPoint inside a group . . . 181

41. Lines style 181
41.0.1. Use of \tkzSetUpLine . . . 181
41.0.2. Change line width . . . 181
41.0.3. Change style of line . . . 182
41.0.4. Example 3: extend lines . . . 182

42. Arc style 182

43. Compass style, configuration macro \tkzSetUpCompass 183
43.0.1. Use of \tkzSetUpCompass . . . 183
43.0.2. Use of \tkzSetUpCompass with \tkzShowLine . . . 184

44. Label style 184

45. Own style 184

IX. Examples 186

46. Some interesting examples 187
46.1. Square root of the integers . . . 187
46.2. About right triangle . . . 187
46.3. Archimedes . . . 188
46.3.1. Square and rectangle of same area; Golden section . . . 189
46.3.2. Steiner Line and Simson Line . . . 190
46.4. Lune of Hippocrates . . . 191
46.5. Lunes of Hasan Ibn al-Haytham . . . 191
46.6. About clipping circles . . . 192
46.7. Similar isosceles triangles . . . 193
46.8. Revised version of ”Tangente” . . . 194
46.9. ”Le Monde” version . . . 195

46.10. Triangle altitudes 196
46.11. Altitudes - other construction 197
46.12. Three circles in an Equilateral Triangle 198
46.13. Law of sines 199
46.14. Flower of Life 200
46.15. Pentagon in a circle 202
46.16. Pentagon in a square 203
46.17. Hexagon Inscribed 204
46.18. Power of a point with respect to a circle 205
46.19. Radical axis of two non-concentric circles 206
46.20. External homothetic center 207
46.21. Tangent lines to two circles 208
46.22. Tangent lines to two circles with radical axis 209
46.23. Definition of a circle _Apollonius_ 210
46.24. Application of Inversion : `Pappus chain` 211
46.25. Book of lemmas proposition 1 Archimedes 212
46.26. Book of lemmas proposition 6 Archimedes 212
46.27. "The" Circle of APOLLONIUS 213

47. Different authors 217
47.1. Code from Andrew Swan 217
47.2. Example: Dimitris Kapeta 217
47.3. Example : John Kitzmiller 218
47.4. Example 1: from Indonesia 219
47.5. Example 2: from Indonesia 220
47.6. Illustration of the Morley theorem by Nicolas François 222
47.7. Gou gu theorem / Pythagorean Theorem by Zhao Shuang 223
47.8. Reuleaux-Triangle 224

X. FAQ 225

48. FAQ 226
48.1. Most common errors 226

Index 228

Part I.

General survey : a brief but comprehensive review

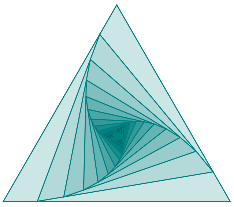
1. Installation

tkz-euclide is on the server of the **CTAN**[1]. If you want to test a beta version, just put the following files in a texmf folder that your system can find. You will have to check several points:

- The **tkz-euclide** folder must be located on a path recognized by **latex**.
- The **tkz-euclide** uses xfp.
- This documentation and all examples were obtained with **lualatex** but **pdflatex** or **xelatex** should be suitable.

1 **tkz-euclide** is part of TeXLive and **tlmgr** allows you to install them. This package is also part of MiKTeX under Windows.

2. Presentation and Overview



```
\begin{tikzpicture}[scale=.25]
\tkzDefPoints{0/0/A,12/0/B,6/12*sind(60)/C}
\foreach \density in {20,30,...,240}{%
  \tkzDrawPolygon[fill=teal!\density](A,B,C)
  \pgfnodealias{X}{A}
  \tkzDefPointWith[linear,K=.15](A,B) \tkzGetPoint{A}
  \tkzDefPointWith[linear,K=.15](B,C) \tkzGetPoint{B}
  \tkzDefPointWith[linear,K=.15](C,X) \tkzGetPoint{C}}
\end{tikzpicture}
```

2.1. Why tkz-euclide?

My initial goal was to provide other mathematics teachers and myself with a tool to quickly create Euclidean geometry figures without investing too much effort in learning a new programming language. Of course, **tkz-euclide** is for math teachers who use LaTeX and makes it possible to easily create correct drawings by means of LaTeX.

It appeared that the simplest method was to reproduce the one used to obtain construction by hand. To describe a construction, you must, of course, define the objects but also the actions that you perform. It seemed to me that syntax close to the language of mathematicians and their students would be more easily understandable; moreover, it also seemed to me that this syntax should be close to that of LaTeX. The objects, of course, are points, segments, lines, triangles, polygons and circles. As for actions, I considered five to be sufficient, namely: define, create, draw, mark and label.

The syntax is perhaps too verbose but it is, I believe, easily accessible. As a result, the students like teachers were able to easily access this tool.

2.2. TikZ vs tkz-euclide

I love programming with Ti*k*Z, and without Ti*k*Z I would never have had the idea to create **tkz-euclide** but never forget that behind it there is Ti*k*Z and that it is always possible to insert code from Ti*k*Z. **tkz-euclide** doesn't prevent you from using Ti*k*Z. That said, I don't think mixing syntax is a good thing.

There is no need to compare Ti*k*Z and **tkz-euclide**. The latter is not addressed to the same audience as Ti*k*Z. The first one allows you to do a lot of things, the second one only does geometry drawings. The first one can do everything the second one does, but the second one will more easily do what you want.

The main purpose is to define points to create geometrical figures. **tkz-euclide** allows you to draw the essential objects of Euclidean geometry from these points but it may be insufficient for some actions like coloring surfaces. In this case you will have to use Ti*k*Z which is always possible.

Here are some comparisons between **Ti*k*Z** and **tkz-euclide** 4. For this I will use the geometry examples from the PGFManual. The two most important Euclidean tools used by early Greeks to construct different geometrical shapes and angles were a compass and a straightedge. My idea is to allow you to follow step by step a construction that would be done by hand (with compass and straightedge) as naturally as possible.

2.2.1. Book I, proposition I _Euclid's Elements_

Book I, proposition I _Euclid's Elements_

To construct an equilateral triangle on a given finite straight line.

Explanation :

The fourth tutorial of the *PgfManual* is about geometric constructions. *T. Tantau* proposes to get the drawing with its beautiful tool Ti*k*Z. Here I propose the same construction with *tkz-elements*. The color of the Ti*k*Z code is green and that of *tkz-elements* is red.

```
\usepackage{tikz}
\usetikzlibrary{calc,intersections,through,backgrounds}
```

```
\usepackage{tkz-euclide}
```

How to get the line AB ? To get this line, we use two fixed points.

```
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (1.25,0.25);
\draw (A) -- (B);
```

```
\tkzDefPoint(0,0){A}
\tkzDefPoint(1.25,0.25){B}
\tkzDrawSegment(A,B)
\tkzLabelPoint[left](A){$A$}
\tkzLabelPoint[right](B){$B$}
```

We want to draw a circle around the points A and B whose radius is given by the length of the line AB.

```
\draw let \p1 = ($ (B) - (A) $),
\n2 = {veclen(\x1,\y1)} in
          (A) circle (\n2)
          (B) circle (\n2);
```

```
\tkzDrawCircles(A,B B,A)
```

The intersection of the circles

```
draw [name path=A--B] (A) -- (B);
node (D) [name path=D,draw,circle through=(B),label=left:$D$] at (A) {};
node (E) [name path=E,draw,circle through=(A),label=right:$E$] at (B) {};
path [name intersections={of=D and E, by={[label=above:$C$]C, [label=below:$C'$]C'}}];
draw [name path=C--C',red] (C) -- (C');
path [name intersections={of=A--B and C--C',by=F}];
node [fill=red,inner sep=1pt,label=-45:$F$] at (F) {};
```

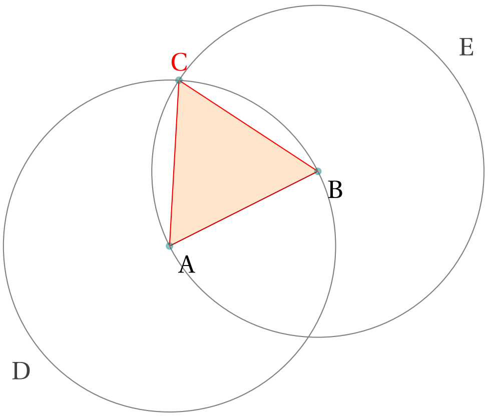
```
\tkzInterCC(A,B)(B,A) \tkzGetPoints{C}{X}
```

How to draw points :

```
\foreach \point in {A,B,C}
\fill [black,opacity=.5] (\point) circle (2pt);
```

```
 \tkzDrawPoints[fill=gray,opacity=.5](A,B,C)
```

2.2.2. Complete code with tkz-euclide




```
\begin{tikzpicture}[scale=1.25,thick,help lines/.style={thin,draw=black!50}]
\tkzDefPoint(0,0){A}
\tkzDefPoint(1.25+rand(),0.25+rand()){B}
\tkzInterCC(A,B)(B,A) \tkzGetPoints{C}{X}

\tkzFillPolygon[triangle,opacity=.5](A,B,C)
\tkzDrawSegment[input](A,B)
\tkzDrawSegments[red](A,C B,C)
\tkzDrawCircles[help lines](A,B B,A)

\tkzLabelPoints(A,B)
\tkzLabelCircle[below=12pt](A,B)(180){$D$}
\tkzLabelCircle[above=12pt](B,A)(180){$E$}
\tkzLabelPoint[above,red](C){$C$}
\tkzDrawPoints[fill=gray,opacity=.5](A,B,C)

\end{tikzpicture}
```

Book I, Proposition II _Euclid's Elements_

Book I, Proposition II _Euclid's Elements_

To place a straight line equal to a given straight line with one end at a given point.

Explanation

In the first part, we need to find the midpoint of the straight line AB. With TikZ we can use the calc library

```
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (1.25,0.25);
\draw (A) -- (B);
\node [fill=red,inner sep=1pt,label=below:$X$] (X) at ($ (A)!.5!(B) $) {};
```

With `tkz-euclide` we have a macro **`\tkzDefMidPoint`**, we get the point X with **`\tkzGetPoint`** but we don't need this point to get the next step.

```
\tkzDefPoints{0/0/A,0.75/0.25/B,1/1.5/C}
\tkzDefMidPoint(A,B) \tkzGetPoint{X}
```

Then we need to construct a triangle equilateral. It's easy with `tkz-euclide` . With TikZ you need some effort because you need to use the midpoint X to get the point D with trigonometry calculation.

```
\node [fill=red,inner sep=1pt,label=below:$X$] (X) at ($ (A)!.5!(B) $) {};
\node [fill=red,inner sep=1pt,label=above:$D$] (D) at
($ (X) ! {sin(60)*2} ! 90:(B) $) {};
\draw (A) -- (D) -- (B);
```

```
\tkzDefTriangle[equilateral](A,B) \tkzGetPoint{D}
```

We can draw the triangle at the end of the picture with

```
\tkzDrawPolygon{A,B,C}
```

We know how to draw the circle around B through C and how to place the points E and F

```
\node (H) [label=135:$H$,draw,circle through=(C)] at (B) {};
\draw (D) -- ($ (D) ! 3.5 ! (B) $) coordinate [label=below:$F$] (F);
\draw (D) -- ($ (D) ! 2.5 ! (A) $) coordinate [label=below:$E$] (E);
```

```
\tkzDrawCircle(B,C)
\tkzDrawLines[add=0 and 2](D,A D,B)
```

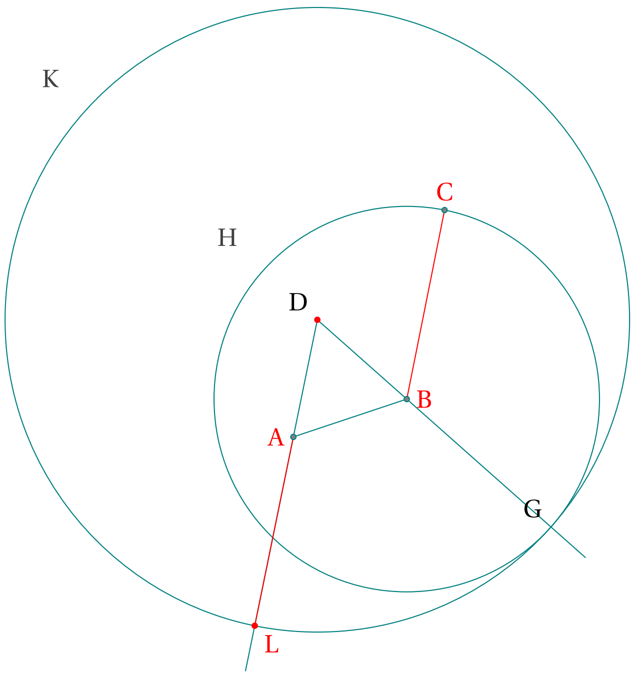
We can place the points E and F at the end of the picture. We don't need them now.
Intersecting a Line and a Circle : here we search the intersection of the circle around B through C and the line DB. The infinite straight line DB intercepts the circle but with TikZ we need to extend the lines DB and that can be done using partway calculations. We get the point F and BF or DF intercepts the circle

```
 \node (H) [label=135:$H$,draw,circle through=(C)] at (B) {};
\path let \p1 = ($ (B) - (C) $) in
  coordinate [label=left:$G$] (G) at ($ (B) ! veclen(\x1,\y1) ! (F) $);
\fill[red,opacity=.5] (G) circle (2pt);
```

Like the intersection of two circles, it's easy to find the intersection of a line and a circle with `elements`. We don't need F

```
\tkzInterLC(B,D)(B,C)\tkzGetFirstPoint{G}
```

there are no more difficulties. Here the final code with some simplications.




```
\begin{tikzpicture}[scale=2]
\tkzDefPoint(0,0){A}
\tkzDefPoint(0.75,0.25){B}
\tkzDefPoint(1,1.5){C}
\tkzDefTriangle[equilateral](A,B) \tkzGetPoint{D}
\tkzInterLC(B,D)(B,C)\tkzGetFirstPoint{G}
\tkzInterLC(D,A)(D,G)\tkzGetSecondPoint{L}
\tkzDrawCircles(B,C D,G)
\tkzDrawLines[add=0 and 2](D,A D,B)
\tkzDrawSegment(A,B)
\tkzDrawSegments[red](A,L B,C)
\tkzDrawPoints[red](D,L)
\tkzDrawPoints[fill=gray](A,B,C)
\tkzLabelPoints[left,red](A)
\tkzLabelPoints[below right,red](L)
\tkzLabelCircle[above left=6pt](B,G)(180){$H$}
\tkzLabelPoints[above left](D,G)
\tkzLabelPoints[above,red](C)
\tkzLabelPoints[right,red](B)
\tkzLabelCircle[above left=6pt](D,G)(180){$K$}
\end{tikzpicture}
```

2.3. `tkz-euclide4` vs `tkz-euclide3`

Now I am no longer a Mathematics teacher, and I only spend a few hours studying geometry. I wanted to avoid multiple complications by trying to make **`tkz-euclide`** independent of **`tkz-base`**. Thus was born **`tkz-euclide`** 4. The latter is a simplified version of its predecessor. The macros of **`tkz-euclide 3`** have been retained. The unit is now **`cm`**. Si vous avez besoin de certaines macros de **`tkz-base`**, il vous faudra sans doute utiliser la macro **`\tkzInit`**.

2.4. How to use the tkz-euclide package ?

2.4.1. Let's look at a classic example

In order to show the right way, we will see how to build an equilateral triangle. Several possibilities are open to us, we are going to follow the steps of Euclid.

- First of all, you have to use a document class. The best choice to test your code is to create a single figure with the class **standalone**.

```
\documentclass{standalone}
```

- Then load the **tkz-euclide** package:

```
\usepackage{tkz-euclide}
```

  You don't need to load Ti*k*Z because the **tkz-euclide** package works on top of TikZ and loads it.

- Start the document and open a TikZ picture environment:

```
\begin{document}
\begin{tikzpicture}
```

- Now we define two fixed points:

```
\tkzDefPoint(0,0){A}
\tkzDefPoint(5,2){B}
```

- Two points define two circles, let's use these circles:

  circle with center A through B and circle with center B through A. These two circles have two points in common.

```
\tkzInterCC(A,B)(B,A)
```

  We can get the points of intersection with

```
\tkzGetPoints{C}{D}
```

- All the necessary points are obtained, we can move on to the final steps including the plots.

```
\tkzDrawCircles[gray,dashed](A,B B,A)
\tkzDrawPolygon(A,B,C)% The triangle
```

- Draw all points A, B, C and D:

```
\tkzDrawPoints(A,...,D)
```

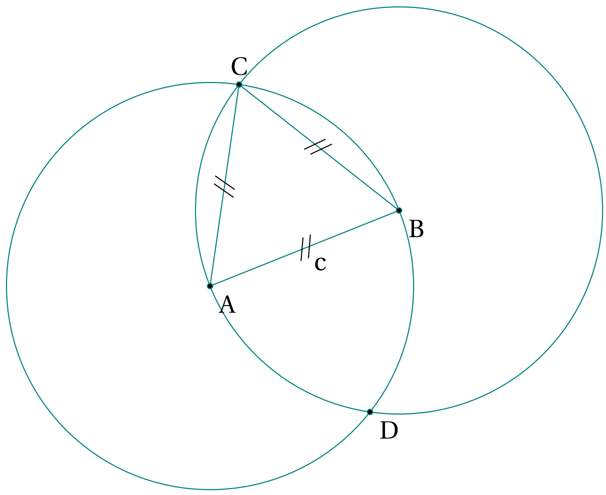
- The final step, we print labels to the points and use options for positioning:

```
\tkzLabelSegments[swap](A,B){$c$}
\tkzLabelPoints(A,B,D)
\tkzLabelPoints[above](C)
```

- We finally close both environments

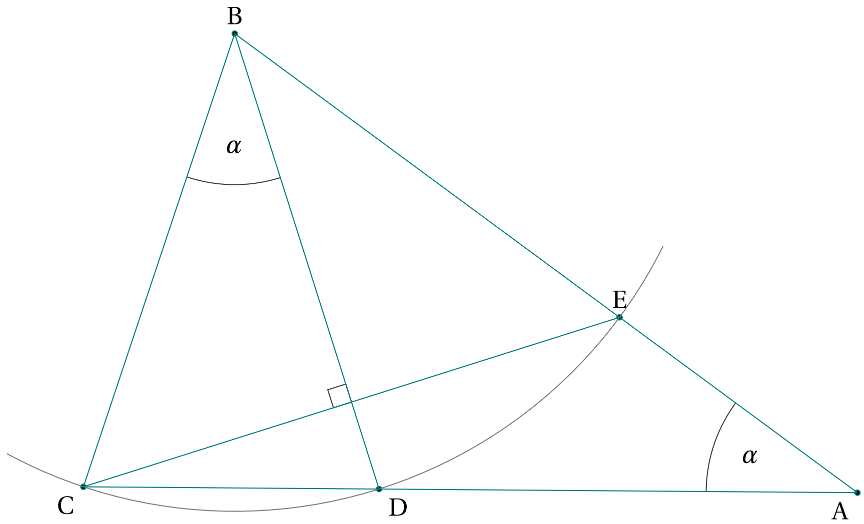
```
\end{tikzpicture}
\end{document}
```

– The complete code




```
\begin{tikzpicture}[scale=.5]
  % fixed points
 \tkzDefPoint(0,0){A}
 \tkzDefPoint(5,2){B}
 % calculus
 \tkzInterCC(A,B)(B,A)
 \tkzGetPoints{C}{D}
 % drawings
 \tkzDrawCircles(A,B B,A)
 \tkzDrawPolygon(A,B,C)
 \tkzDrawPoints(A,...,D)
 % marking
 \tkzMarkSegments[mark=s||](A,B B,C C,A)
 % labelling
 \tkzLabelSegments[swap](A,B){$c$}
 \tkzLabelPoints(A,B,D)
 \tkzLabelPoints[above](C)
\end{tikzpicture}
```

2.4.2. Part I: golden triangle



Let's analyze the figure

1. CBD and DBE are isosceles triangles;
2. BC = BE and (BD) is a bisector of the angle CBE;
3. From this we deduce that the CBD and DBE angles are equal and have the same measure α

$$\widehat{\mathrm{BAC}}+\widehat{\mathrm{ABC}}+\widehat{\mathrm{BCA}}=180^\circ \text{ in the triangle BAC}$$

$$3\alpha+\widehat{\mathrm{BCA}}=180^\circ \text{ in the triangle CBD}$$

then

$$\alpha+2\widehat{\mathrm{BCA}}=180^\circ$$

or

$$\widehat{\mathrm{BCA}}=90^\circ-\alpha/2$$

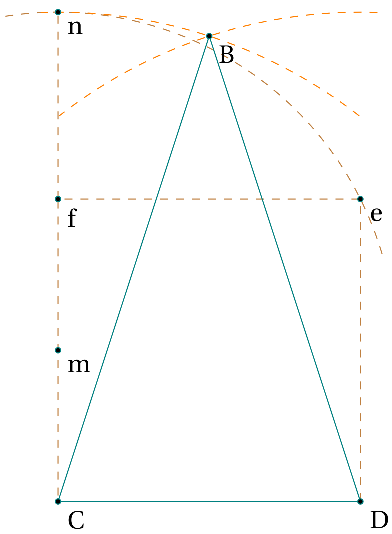
4. Finally

$$\widehat{CBD} = \alpha = 36^\circ$$

the triangle CBD is a "gold" triangle.

How construct a gold triangle or an angle of 36°?

1. We place the fixed points C and D. `\tkzDefPoint(0,0){C}` and `\tkzDefPoint(4,0){D}`;
2. We construct a square CDef and we construct the midpoint m of [Cf];
   We can do all of this with a compass and a rule;
3. Then we trace an arc with center m through e. This arc cross the line (Cf) at n;
4. Now the two arcs with center C and D and radius Cn define the point B.




```
\begin{tikzpicture}
\tkzDefPoint(0,0){C}
\tkzDefPoint(4,0){D}
\tkzDefSquare(C,D)
\tkzGetPoints{e}{f}
\tkzDefMidPoint(C,f)
\tkzGetPoint{m}
\tkzInterLC(C,f)(m,e)
\tkzGetSecondPoint{n}
\tkzInterCC[with nodes](C,C,n)(D,C,n)
\tkzGetFirstPoint{B}
\tkzDrawSegment[brown,dashed](f,n)
\tkzDrawPolygon[brown,dashed](C,D,e,f)
\tkzDrawArc[brown,dashed](m,e)(n)
\tkzCompass[brown,dashed,delta=20](C,B)
\tkzCompass[brown,dashed,delta=20](D,B)
\tkzDrawPoints(C,D,B)
\tkzDrawPolygon(B,...,D)
\end{tikzpicture}
```

After building the golden triangle BCD, we build the point A by noticing that BD = DA. Then we get the point E and finally the point F. This is done with already intersections of defined objects (line and circle).

```
\begin{tikzpicture}
  \tkzDefPoint(0,0){C}
  \tkzDefPoint(4,0){D}
  \tkzDefSquare(C,D)
  \tkzGetPoints{e}{f}
  \tkzDefMidPoint(C,f)
  \tkzGetPoint{m}
  \tkzInterLC(C,f)(m,e)
  \tkzGetSecondPoint{n}
  \tkzInterCC[with nodes](C,C,n)(D,C,n)
  \tkzGetFirstPoint{B}
  \tkzInterLC(C,D)(D,B) \tkzGetSecondPoint{A}
  \tkzInterLC(B,A)(B,D) \tkzGetSecondPoint{E}
  \tkzInterLL(B,D)(C,E) \tkzGetPoint{F}
  \tkzDrawPoints(C,D,B)
  \tkzDrawPolygon(B,...,D)
```

```
  \tkzDrawPolygon(B,C,D)
  \tkzDrawSegments(D,A A,B C,E)
  \tkzDrawArc[delta=10](B,C)(E)
  \tkzDrawPoints(A,...,F)
  \tkzMarkRightAngle(B,F,C)
  \tkzMarkAngles(C,B,D E,A,D)
  \tkzLabelAngles[pos=1.5](C,B,D E,A,D){$\alpha$}
  \tkzLabelPoints[below](A,C,D,E)
  \tkzLabelPoints[above right](B,F)
\end{tikzpicture}
```

2.4.3. Part II: two others methods with golden and euclid triangle

tkz-euclide knows how to define a ”golden” or ”euclide” triangle. We can define BCD and BCA like gold triangles.

```
\begin{tikzpicture}
  \tkzDefPoint(0,0){C}
  \tkzDefPoint(4,0){D}
  \tkzDefTriangle[euclid](C,D)
  \tkzGetPoint{B}
  \tkzDefTriangle[euclid](B,C)
  \tkzGetPoint{A}
  \tkzInterLC(B,A)(B,D) \tkzGetSecondPoint{E}
  \tkzInterLL(B,D)(C,E) \tkzGetPoint{F}
  \tkzDrawPoints(C,D,B)
  \tkzDrawPolygon(B,...,D)
  \tkzDrawPolygon(B,C,D)
  \tkzDrawSegments(D,A A,B C,E)
  \tkzDrawArc[delta=10](B,C)(E)
  \tkzDrawPoints(A,...,F)
  \tkzMarkRightAngle(B,F,C)
  \tkzMarkAngles(C,B,D E,A,D)
  \tkzLabelAngles[pos=1.5](C,B,D E,A,D){$\alpha$}
  \tkzLabelPoints[below](A,C,D,E)
  \tkzLabelPoints[above right](B,F)
\end{tikzpicture}
```

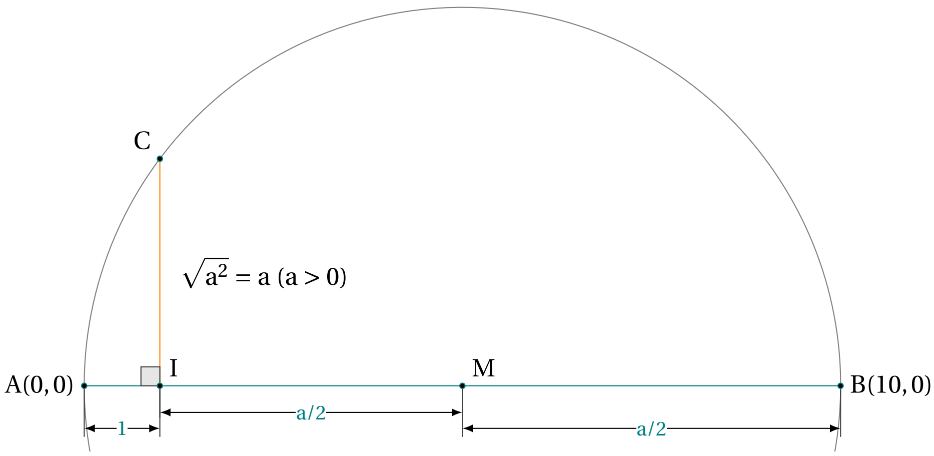
Here is a final method that uses rotations:

```
\begin{tikzpicture}
\tkzDefPoint(0,0){C} % possible
% \tkzDefPoint[label=below:$C$](0,0){C}
% but don't do this
\tkzDefPoint(2,6){B}
% We get D and E with a rotation
\tkzDefPointBy[rotation= center B angle 36](C) \tkzGetPoint{D}
\tkzDefPointBy[rotation= center B angle 72](C) \tkzGetPoint{E}
% To get A we use an intersection of lines
\tkzInterLL(B,E)(C,D) \tkzGetPoint{A}
\tkzInterLL(C,E)(B,D) \tkzGetPoint{H}
% drawing
\tkzDrawArc[delta=10](B,C)(E)
\tkzDrawPolygon(C,B,D)
\tkzDrawSegments(D,A B,A C,E)
```

```
% angles
\tkzMarkAngles(C,B,D E,A,D) %this is to draw the arcs
\tkzLabelAngles[pos=1.5](C,B,D E,A,D){$\alpha$}
\tkzMarkRightAngle(B,H,C)
\tkzDrawPoints(A,...,E)
% Label only now
\tkzLabelPoints[below left](C,A)
\tkzLabelPoints[below right](D)
\tkzLabelPoints[above](B,E)
\end{tikzpicture}
```

2.4.4. Complete but minimal example

A unit of length being chosen, the example shows how to obtain a segment of length $\sqrt{a}$ from a segment of length a, using a ruler and a compass.
IB = a, AI = 1



Comments

- The Preamble

  Let us first look at the preamble. If you need it, you have to load **xcolor** before **tkz-euclide**, that is, before TikZ. Ti*k*Z may cause problems with the active characters, but... provides a library in its latest version that's supposed to solve these problems `babel`.

```
\documentclass{standalone} % or another class
   % \usepackage{xcolor} % before tikz or tkz-euclide if necessary
\usepackage{tkz-euclide} % no need to load TikZ
   % \usetkzobj{all}  is no longer necessary
   % \usetikzlibrary{babel}  if there are problems with the active characters
```

  The following code consists of several parts:

- Definition of fixed points: the first part includes the definitions of the points necessary for the construction, these are the fixed points. The macros `\tkzInit` and `\tkzClip` in most cases are not necessary.

```
\tkzDefPoint(0,0){A}
\tkzDefPoint(1,0){I}
```

- The second part is dedicated to the creation of new points from the fixed points; a B point is placed at 10 cm from A. The middle of [AB] is defined by M and then the orthogonal line to the (AB) line is searched for at the I point. Then we look for the intersection of this line with the semi-circle of center M passing through A.

```
\tkzDefPointBy[homothety=center A ratio  10 ](I)
   \tkzGetPoint{B}
\tkzDefMidPoint(A,B)
   \tkzGetPoint{M}
\tkzDefPointWith[orthogonal](I,M)
   \tkzGetPoint{H}
\tkzInterLC(I,H)(M,B)
\tkzGetSecondPoint{C}
```

– The third one includes the different drawings;

```
\tkzDrawSegment[style=orange](I,H)
\tkzDrawPoints(O,I,A,B,M)
\tkzDrawArc(M,A)(O)
\tkzDrawSegment[dim={$1$,-16pt,}](A,I)
\tkzDrawSegment[dim={$a/2$,-10pt,}](I,M)
\tkzDrawSegment[dim={$a/2$,-16pt,}](M,B)
```

– Marking: the fourth is devoted to marking;

```
\tkzMarkRightAngle[ra](A,I,C)
```

– Labelling: the latter only deals with the placement of labels.

```
\tkzLabelPoint[left](A){$A(0,0)$}
\tkzLabelPoint[right](B){$B(10,0)$}
\tkzLabelSegment[right=4pt](I,C){$\sqrt{a^2}=a \ (a>0)$}
```

– The full code:

```
\begin{tikzpicture}[scale=1,ra/.style={fill=gray!20}]
   % fixed points
   \tkzDefPoint(0,0){A}
   \tkzDefPoint(1,0){I}
   % calculation
   \tkzDefPointBy[homothety=center A ratio  10 ](I) \tkzGetPoint{B}
   \tkzDefMidPoint(A,B)              \tkzGetPoint{M}
   \tkzDefPointWith[orthogonal](I,M) \tkzGetPoint{H}
   \tkzInterLC(I,H)(M,B)             \tkzGetSecondPoint{C}
   \tkzDrawSegment[style=orange](I,C)
   \tkzDrawArc(M,B)(A)
   \tkzDrawSegment[dim={$1$,-16pt,}](A,I)
   \tkzDrawSegment[dim={$a/2$,-10pt,}](I,M)
   \tkzDrawSegment[dim={$a/2$,-16pt,}](M,B)
   \tkzMarkRightAngle[ra](A,I,C)
   \tkzDrawPoints(I,A,B,C,M)
   \tkzLabelPoint[left](A){$A(0,0)$}
   \tkzLabelPoints[above right](I,M)
   \tkzLabelPoints[above left](C)
   \tkzLabelPoint[right](B){$B(10,0)$}
```

```
    \tkzLabelSegment[right=4pt](I,C){$\sqrt{a^2}=a \ (a>0)$}
\end{tikzpicture}
```

3. The Elements of tkz code

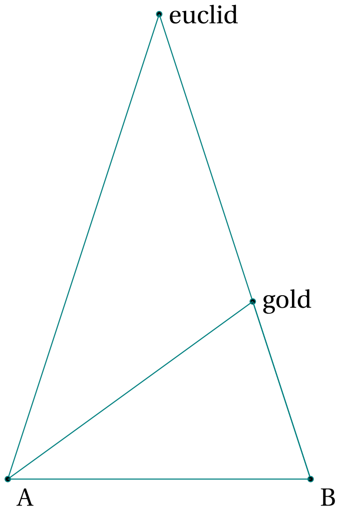
To work with my package, you need to have notions of LaTeX as well as TikZ.
In this paragraph, we start looking at the "rules" and "symbols" used to create a figure with **tkz-euclide**.

3.1. Objects and language

The primitive objects are points. You can refer to a point at any time using the name given when defining it. (it is possible to assign a different name later on).
To get new points you will use macros. **tkz-euclide** macros have a name beginning with tkz. There are four main categories starting with: `\tkzDef...` `\tkzDraw...` `\tkzMark...` and `\tkzLabel...`. The used points are passed as parameters between parentheses while the created points are between braces.
Le code des figures est placés dans un environnement **tikzpicture**

```
\begin{tikzpicture}
  code ...
\end{tikzpicture}
```

Contrary to TikZ, you should not end a macro with ";". We thus lose the important notion which is the **path**. However, it is possible to place some code between the macros **tkz-euclide**.
Among the first category, `\tkzDefPoint` allows you to define fixed points. It will be studied in detail later. Here we will see in detail the macro `\tkzDefTriangle`.
This macro makes it possible to associate to a pair of points a third point in order to define a certain triangle `\tkzDefTriangle(A,B)`. The obtained point is referenced `tkzPointResult` and it is possible to choose another reference with `\tkzGetPoint{C}` for example.
`\tkzDefTriangle[euclid](A,B) \tkzGetPoint{C}`
Parentheses are used to pass arguments. In `(A,B)` A and B are the points with which a third will be defined. However, in `{C}` we use braces to retrieve the new point.
In order to choose a certain type of triangle among the following choices: `equilateral`, `isosceles right`, `half`, `pythagoras`, `school`, `golden or sublime`, `euclid`, `gold`, `cheops`... and `two angles` you just have to choose between hooks, for example:




```
\begin{tikzpicture}[scale=.5]
\tkzDefPoints{0/0/A,8/0/B}
\foreach \tr in {euclid,gold}
{\tkzDefTriangle[\tr](A,B) \tkzGetPoint{C}
\tkzDrawPoint(C)
\tkzLabelPoint[right](C){\tr}
\tkzDrawSegments(A,C C,B)}
\tkzDrawPoints(A,B)
\tkzDrawSegments(A,B)
\tkzLabelPoints(A,B)
\end{tikzpicture}
```

3.2. Notations and conventions

I deliberately chose to use the geometric French and personal conventions to describe the geometric objects represented. The objects defined and represented by **tkz-euclide** are points, lines and circles located in a plane. They are the primary objects of Euclidean geometry from which we will construct figures.

According to **Euclid**, these figures will only illustrate pure ideas produced by our brain. Thus a point has no dimension and therefore no real existence. In the same way the line has no width and therefore no existence in the real world. The objects that we are going to consider are only representations of ideal mathematical objects. `tkz-euclide` will follow the steps of the ancient Greeks to obtain geometrical constructions using the ruler and the compass.
Here are the notations that will be used:

– The points are represented geometrically either by a small disc or by the intersection of two lines (two straight lines, a straight line and a circle or two circles). In this case, the point is represented by a cross.


```
\begin{tikzpicture}
  \tkzDefPoints{0/0/A,4/2/B}
  \tkzDrawPoints(A,B)
  \tkzLabelPoints(A,B)
\end{tikzpicture}
```

or else


```
\begin{tikzpicture}
  \tkzSetUpPoint[shape=cross, color=red]
  \tkzDefPoints{0/0/A,4/2/B}
  \tkzDrawPoints(A,B)
  \tkzLabelPoints(A,B)
  \end{tikzpicture}
```

The existence of a point being established, we can give it a label which will be a capital letter (with some exceptions) of the Latin alphabet such as A, B or C. For example:

- O is a center for a circle, a rotation, etc.;
- M defined a midpoint;
- H defined the foot of an altitude;
- P' is the image of P by a transformation ;

It is important to note that the reference name of a point in the code may be different from the label to designate it in the text. So we can define a point A and give it as label P. In particular the style will be different, point A will be labeled A.


```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDrawPoints(A)
  \tkzLabelPoint(A){$P$}
\end{tikzpicture}
```

Exceptions: some points such as the middle of the sides of a triangle share a characteristic, so it is normal that their names also share a common character. We will designate these points by M_a, M_b and M_c or M_A, M_B and M_C.

In the code, these points will be referred to as: M_A, M_B and M_C.

Another exception relates to intermediate construction points which will not be labelled. They will often be designated by a lowercase letter in the code.

– The line segments are designated by two points representing their ends in square brackets: [AB].

– The straight lines are in Euclidean geometry defined by two points so A and B define the straight line (AB). We can also designate this stright line using the Greek alphabet and name it (δ) or (Δ). It is also possible to designate the straight line with lowercase letters such as d and d'.

- The semi-straight line is designated as follows [AB).
- Relation between the straight lines. Two perpendicular (AB) and (CD) lines will be written $(AB) \perp (CD)$ and if they are parallel we will write $(AB) \mathbin{/\!/} (CD)$.
- The lengths of the sides of triangle ABC are AB, AC and BC. The numbers are also designated by a lowercase letter so we will write: AB = c, AC = b and BC = a. The letter a is also used to represent an angle, and r is frequently used to represent a radius, d a diameter, l a length, d a distance.
- Polygons are designated afterwards by their vertices so ABC is a triangle, EFGH a quadrilateral.
- Angles are generally measured in degrees (ex 60°) and in an equilateral ABC triangle we will write $\widehat{ABC} = \widehat{B} = 60^\circ$.
- The arcs are designated by their extremities. For example if A and B are two points of the same circle then $\overparen{AB}$.
- Circles are noted either $\mathscr{C}$ if there is no possible confusion or $\mathscr{C}(O\,;\,A)$ for a circle with center O and passing through the point A or $\mathscr{C}(O\,;\,1)$ for a circle with center O and radius 1 cm.
- Name of the particular lines of a triangle: I used the terms bisector, bisector out, mediator (sometimes called perpendicular bisectors), altitude, median and symmedian.
- (x_1,y_1) coordinates of the point A_1, (x_A,y_A) coordinates of the point A.

3.3. Set, Calculate, Draw, Mark, Label

The title could have been: `Separation of Calculus and Drawings`
When a document is prepared using the LaTeX system, the source code of the document can be divided into two parts: the document body and the preamble. Under this methodology, publications can be structured, styled and typeset with minimal effort. I propose a similar methodology for creating figures with **`tkz-euclide`**.
The first part defines the fixed points, the second part allows the creation of new points. **`Set and Calculate`** are the two main parts. All that is left to do is to draw (or fill), mark and label. It is possible that **`tkz-euclide`** is insufficient for some of these latter actions but you can use Ti*k*Z
One last remark that I think is important, it is best to avoid introducing coordinates within a code as much as possible. I think that the coordinates should appear at the beginning of the code with the fixed points. Then the use of references is recommended. Most macros have the option **`nodes`** or **`with nodes`**.
I also think it's best to define the styles of the different objects from the beginning.

4. News and compatibility

Some changes have been made to make the syntax more homogeneous and especially to distinguish the definition and search for coordinates from the rest, i.e. drawing, marking and labelling. In the future, the definition macros being isolated, it will be easier to introduce a phase of coordinate calculations using **Lua**.
Here are some of the changes. I'm sorry but the list of changes and novelties is made in the greatest disorder!

- An important novelty is the recent replacement of the `fp` package by `xfp`. This is to improve the calculations a little bit more and to make it easier to use;
- Improved code and bug fixes;
- First of all, you don't have to deal with Tik Z the size of the bounding box. Early versions of **tkz-euclide** did not control the size of the bounding box, The bounding box is now controlled in each macro (hopefully) to avoid the use of `\tkzInit` followed by `\tkzClip`;
- With **tkz-euclide** loads all objects, so there's no need to place `\usetkzobj{all}`;
- Added macros for the bounding box: `\tkzSaveBB` `\tkzClipBB` and so on;
- Logically most macros accept Ti*k*Z options. So I removed the "duplicate" options when possible thus the "label options" option is removed;
- The unit is now the cm;
- `\tkzCalcLength` `\tkzGetLength` gives result in cm;
- `\tkzMarkArc` and `\tkzLabelArc` are new macros;
- Now `\tkzClipCircle` and `\tkzClipPolygon` have an option **out**. To use this option you must have a Bounding Box that contains the object on which the Clip action will be performed. Cela peut se faire en utilisant un objet qui englobe la figure ou bien en utilisant la macro `\tkzInit`;
- The options **end** and **start** which allowed to give a label to a straight line are removed. You now have to use the macro `\tkzLabelLine`;
- Introduction of the libraries `quotes` and `angles`; it allows to give a label to a point, even if I am not in favour of this practice;
- The notion of vector disappears, to draw a vector just pass "->" as an option to `\tkzDrawSegment`;
- `\tkzDrawMedian`, `\tkzDrawBisector`, `\tkzDrawAltitude`, `\tkzDrawMedians`, `\tkzDrawBisectors` et `\tkzDrawAltitudes` do not exist anymore. The creation and drawing separation is not respected so it is preferable to first create the coordinates of these points with `\tkzDefSpcTriangle[median]` and then to choose the ones you are going to draw with `\tkzDrawSegments` or `\tkzDrawLines`;
- `\tkzDefIntSimilitudeCenter` and `\tkzDefExtSimilitudeCenter` do not exist anymore;
- `\tkzDrawTriangle` has been deleted. `\tkzDrawTriangle[equilateral]` was handy but it is better to get the third point with `\tkzDefTriangle[equilateral]` and then draw with `\tkzDrawPolygon`; idem for `\tkzDrawSquare` and `\tkzDrawGoldRectangle`;

– \tkzDefRandPointOn is replaced by \tkzGetRandPointOn;

– now \tkzTangent is replaced by \tkzDefTangent;

– An option of the macro **\tkzDefTriangle** has changed, in the previous version the option was ”euclide” with an ”e”. Now it’s ”euclid”;

– Random points are now in **tkz-euclide** and the macro **\tkzGetRandPointOn** is replaced by **\tkzDefRandPointOn**. For homogeneity reasons, the points must be retrieved with **\tkzGetPoint**;

– New macros have been added : **\tkzDrawSemiCircles**, **\tkzDrawPolygons**, **\tkzDrawTriangles**;

– Option ”isosceles right” is a new option of the macro **\tkzDefTriangle**;

– Appearance of the macro **\usetkztool** which allows to load new ”tools”;

– The styles can be modified with the help of the following macros : **\tkzSetUpPoint**, **\tkzSetUpLine**, **\tkzSetUpArc**, **\tkzSetUpCompass**, **\tkzSetUpLabel** and **\tkzSetUpStyle**. The last one allows you to create a new style.

Part II.

Setting

5. First step: fixed points

The first step in a geometric construction is to define the fixed points from which the figure will be constructed. The general idea is to avoid manipulating coordinates and to prefer to use the references of the points fixed in the first step or obtained using the tools provided by the package. Even if it's possible, I think it's a bad idea to work directly with coordinates. Preferable is to use named points.
tkz-euclide uses macros and vocabulary specific to geometric construction. It is of course possible to use the tools of Ti*k*Z but it seems more logical to me not to mix the different syntaxes.
A point in **tkz-euclide** is a particular "node" for Ti*k*Z. In the next section we will see how to define points using coordinates. The style of the points (color and shape) will not be discussed. You will find some indications in some examples; for more information you can read the following section 39.

6. Definition of a point : \tkzDefPoint or \tkzDefPoints

Points can be specified in any of the following ways:

– Cartesian coordinates;

– Polar coordinates;

– Named points;

– Relative points.

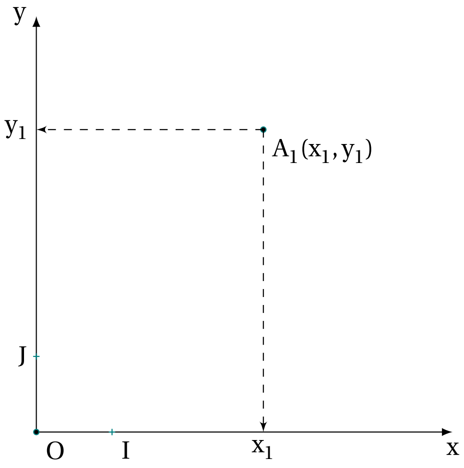
A point is defined if it has a name linked to a unique pair of decimal numbers. Let (x, y) or $(a{:}\,d)$ i.e. (x abscissa, y ordinate) or (a angle: d distance). This is possible because the plan has been provided with an orthonormed Cartesian coordinate system. The working axes are (ortho)normed with unity equal to 1 cm.
The Cartesian coordinate (a, b) refers to the point a centimeters in the x-direction and b centimeters in the y-direction.
A point in polar coordinates requires an angle α, in degrees, and a distance d from the origin with a dimensional unit by default it's the `cm`.
The `\tkzDefPoint` macro is used to define a point by assigning coordinates to it. This macro is based on `\coordinate`, a macro of Ti*k*Z. It can use Ti*k*Z-specific options such as **shift**. If calculations are required then the `xfp` package is chosen. We can use Cartesian or polar coordinates.

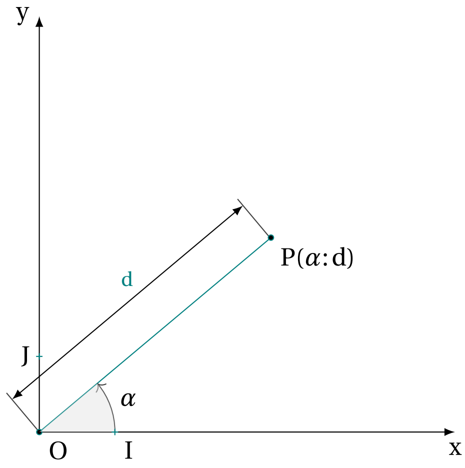
Cartesian coordinates

```
\begin{tikzpicture}[scale=1]
  \tkzInit[xmax=5,ymax=5]
  % necessary to limit
  % the size of the axes
  \tkzDrawX[>=latex]
  \tkzDrawY[>=latex]
  \tkzDefPoints{0/0/O,1/0/I,0/1/J}
  \tkzDefPoint(3,4){A}
  \tkzDrawPoints(O,A)
  \tkzLabelPoint(A){$A_1 (x_1,y_1)$}
  \tkzShowPointCoord[xlabel=$x_1$,
                     ylabel=$y_1$](A)
  \tkzLabelPoints(O,I)
  \tkzLabelPoints[left](J)
  \tkzDrawPoints[shape=cross](I,J)
\end{tikzpicture}
```

Polar coordinates

```
\begin{tikzpicture}[,scale=1]
  \tkzInit[xmax=5,ymax=5]
  \tkzDrawX[>=latex]
  \tkzDrawY[>=latex]
  \tkzDefPoints{0/0/O,1/0/I,0/1/J}
  \tkzDefPoint(40:4){P}
  \tkzDrawSegment[dim={$d$,
                 16pt,above=6pt}](O,P)
  \tkzDrawPoints(O,P)
  \tkzMarkAngle[mark=none,->](I,O,P)
  \tkzFillAngle[opacity=.5](I,O,P)
  \tkzLabelAngle[pos=1.25](I,O,P){%
                             $\alpha$}
  \tkzLabelPoint(P){$P  (\alpha : d )$}
  \tkzDrawPoints[shape=cross](I,J)
  \tkzLabelPoints(O,I)
  \tkzLabelPoints[left](J)
\end{tikzpicture}
```





6.1. Defining a named point \tkzDefPoint

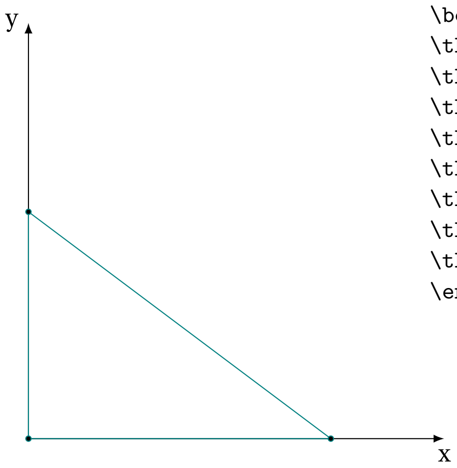
\tkzDefPoint[⟨local options⟩](⟨x,y⟩){⟨ref⟩} or (⟨α:d⟩){⟨ref⟩}

arguments	default	definition
(x,y)	no default	x and y are two dimensions, by default in cm.
(α:d)	no default	α is an angle in degrees, d is a dimension
{ref}	no default	Reference assigned to the point: A, T_a ,P1 or P_1

The obligatory arguments of this macro are two dimensions expressed with decimals, in the first case they are two measures of length, in the second case they are a measure of length and the measure of an angle in degrees. Do not confuse the reference with the name of a point. The reference is used by calculations, but frequently, the name is identical to the reference.

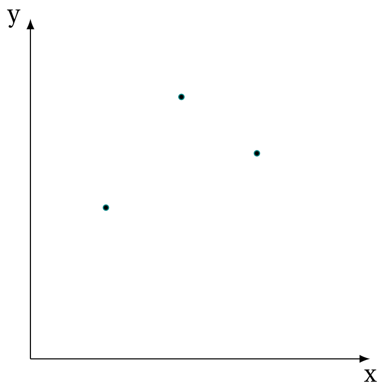
options	default	definition
label	no default	allows you to place a label at a predefined distance
shift	no default	adds (x,y) or (α:d) to all coordinates

6.1.1. Cartesian coordinates



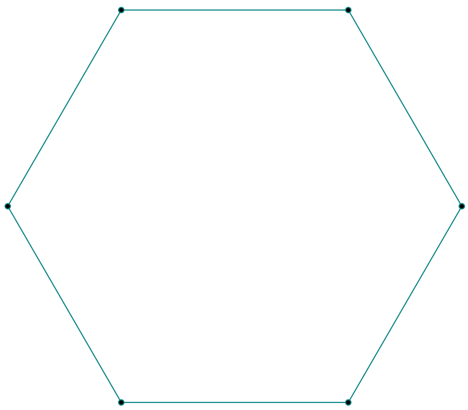

```
\begin{tikzpicture}
\tkzInit[xmax=5,ymax=5] % limits the size of the axes
\tkzDrawX[>=latex]
\tkzDrawY[>=latex]
\tkzDefPoint(0,0){A}
\tkzDefPoint(4,0){B}
\tkzDefPoint(0,3){C}
\tkzDrawPolygon(A,B,C)
\tkzDrawPoints(A,B,C)
\end{tikzpicture}
```

6.1.2. Calculations with xfp




```
\begin{tikzpicture}[scale=1]
  \tkzInit[xmax=4,ymax=4]
  \tkzDrawX\tkzDrawY
  \tkzDefPoint(-1+2,sqrt(4)){O}
  \tkzDefPoint({3*ln(exp(1))},{exp(1)}){A}
  \tkzDefPoint({4*sin(pi/6)},{4*cos(pi/6)}){B}
  \tkzDrawPoints(O,B,A)
\end{tikzpicture}
```

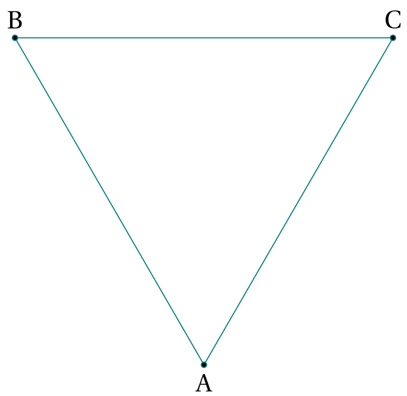
6.1.3. Polar coordinates



```
  \begin{tikzpicture}
  \foreach \an [count=\i] in {0,60,...,300}
   { \tkzDefPoint(\an:3){A_\i}}
  \tkzDrawPolygon(A_1,A_...,A_6)
  \tkzDrawPoints(A_1,A_...,A_6)
  \end{tikzpicture}
```

6.1.4. Relative points

First, we can use the **scope** environment from Ti*k*Z. In the following example, we have a way to define an equilateral triangle.




```
\begin{tikzpicture}[scale=1]
 \begin{scope}[rotate=30]
  \tkzDefPoint(2,3){A}
  \begin{scope}[shift=(A)]
     \tkzDefPoint(90:5){B}
     \tkzDefPoint(30:5){C}
  \end{scope}
 \end{scope}
 \tkzDrawPolygon(A,B,C)
\tkzLabelPoints[above](B,C)
\tkzLabelPoints[below](A)
\tkzDrawPoints(A,B,C)
\end{tikzpicture}
```

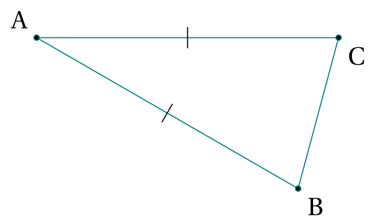
6.2. Point relative to another: \tkzDefShiftPoint

\tkzDefShiftPoint[⟨Point⟩](⟨x,y⟩){⟨ref⟩} or (⟨α:d⟩){⟨ref⟩}

arguments	default	definition
(x,y)	no default	x and y are two dimensions, by default in cm.
(α:d)	no default	α is an angle in degrees, d is a dimension
{ref}	no default	Reference assigned to the point: A, T_a ,P1 or P_1
options	**default**	**definition**
[pt]	no default	\tkzDefShiftPoint[A](0:4){B}

6.2.1. Isosceles triangle

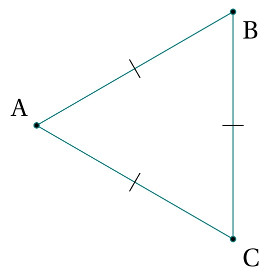
This macro allows you to place one point relative to another. This is equivalent to a translation. Here is how to construct an isosceles triangle with main vertex A and angle at vertex of 30°.




```
\begin{tikzpicture}[rotate=-30]
 \tkzDefPoint(2,3){A}
 \tkzDefShiftPoint[A](0:4){B}
 \tkzDefShiftPoint[A](30:4){C}
 \tkzDrawSegments(A,B B,C C,A)
 \tkzMarkSegments[mark=|](A,B A,C)
 \tkzDrawPoints(A,B,C)
 \tkzLabelPoints(B,C)
 \tkzLabelPoints[above left](A)
\end{tikzpicture}
```

6.2.2. Equilateral triangle

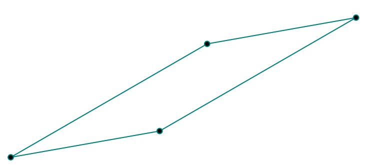
Let's see how to get an equilateral triangle (there is much simpler)




```
\begin{tikzpicture}[scale=1]
 \tkzDefPoint(2,3){A}
 \tkzDefShiftPoint[A](30:3){B}
 \tkzDefShiftPoint[A](-30:3){C}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawPoints(A,B,C)
 \tkzLabelPoints(B,C)
 \tkzLabelPoints[above left](A)
 \tkzMarkSegments[mark=|](A,B A,C B,C)
\end{tikzpicture}
```

6.2.3. Parallelogram

There's a simpler way



```
\begin{tikzpicture}
 \tkzDefPoint(0,0){A}
 \tkzDefPoint(30:3){B}
 \tkzDefShiftPointCoord[B](10:2){C}
 \tkzDefShiftPointCoord[A](10:2){D}
 \tkzDrawPolygon(A,...,D)
 \tkzDrawPoints(A,...,D)
\end{tikzpicture}
```

6.3. Definition of multiple points: \tkzDefPoints

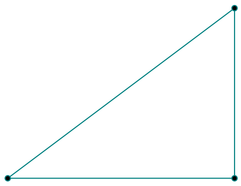
`\tkzDefPoints[⟨local options⟩]{⟨x1/y1/n1,x2/y2/r2, ...⟩}`

x_i and y_i are the coordinates of a referenced point r_i

arguments	default	example
$x_i/y_i/r_i$		`\tkzDefPoints{0/0/O,2/2/A}`

options	default	definition
shift	no default	Adds (x,y) or (α:d) to all coordinates

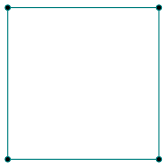
6.4. Create a triangle



```
\begin{tikzpicture}[scale=.75]
 \tkzDefPoints{0/0/A,4/0/B,4/3/C}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawPoints(A,B,C)
\end{tikzpicture}
```

6.5. Create a square

Note here the syntax for drawing the polygon.



```
\begin{tikzpicture}[scale=1]
 \tkzDefPoints{0/0/A,2/0/B,2/2/C,0/2/D}
 \tkzDrawPolygon(A,...,D)
 \tkzDrawPoints(A,...,D)
\end{tikzpicture}
```

Part III.

Calculating

Now that the fixed points are defined, we can with their references using macros from the package or macros that you will create get new points. The calculations may not be apparent but they are usually done by the package. Vous aurez peut-être besoin d'utiliser certains constantes mathématiques, voici la liste des constantes définies par le package.

7. Auxiliary tools

7.1. Constants

tkz-euclide knows some constants, here is the list:

```
\def\tkzPhi{1.618034}
\def\tkzInvPhi{0.618034}
\def\tkzSqrtPhi{1.27202}
\def\tkzSqrTwo{1.414213}
\def\tkzSqrThree{1.7320508}
\def\tkzSqrFive{2.2360679}
\def\tkzSqrTwobyTwo{0.7071065}
\def\tkzPi{3.1415926}
\def\tkzEuler{2.71828182}
```

7.2. New point by calculation

When a macro of **tkznameofpack** creates a new point, it is stored internally with the reference **tkzPointResult**. You can assign your own reference to it. This is done with the macro **\tkzGetPoint**. A new reference is created, your choice of reference must be placed between braces.

\tkzGetPoint{⟨ref⟩}

If the result is in **tkzPointResult**, you can access it with **\tkzGetPoint**.

arguments	default	example	
ref	no default	\tkzGetPoint{M}	see the next example

Sometimes you need to get two points. It's possible with

\tkzGetPoints{⟨ref1⟩}{⟨ref2⟩}

The result is in **tkzPointFirstResult** and **tkzPointSecondResult**.

arguments	default	example	
{ref1,ref2}	no default	\tkzGetPoints{M,N}	It's the case with \tkzInterCC

If you need only the first or the second point you can also use :

\tkzGetFirstPoint{⟨ref1⟩}

arguments	default	example
ref1	no default	\tkzGetFirstPoint{M}

`\tkzGetSecondPoint{⟨ref2⟩}`

arguments	default	example
ref2	no default	\tkzGetSecondPoint{M}

Parfois les résultats consistent en un point et une dimension. Vous obtenez le point avec **\tkzGetPoint** et la dimension avec **\tkzGetLength**.

`\tkzGetLength{⟨name of a macro⟩}`

arguments	default	example
name of a macro	no default	\tkzGetLength{rAB} \rAB gives the length in cm

8. Special points

Here are some special points.

8.1. Middle of a segment \tkzDefMidPoint

It is a question of determining the middle of a segment.

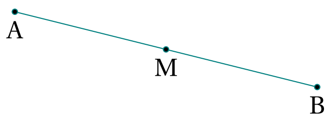
`\tkzDefMidPoint(⟨pt1,pt2⟩)`

The result is in **tkzPointResult**. We can access it with **\tkzGetPoint**.

arguments	default	definition
(pt1,pt2)	no default	pt1 and pt2 are two points

8.1.1. Use of \tkzDefMidPoint

Review the use of **\tkzDefPoint**.




```
\begin{tikzpicture}[scale=1]
 \tkzDefPoint(2,3){A}
 \tkzDefPoint(6,2){B}
 \tkzDefMidPoint(A,B)
 \tkzGetPoint{M}
 \tkzDrawSegment(A,B)
 \tkzDrawPoints(A,B,M)
 \tkzLabelPoints[below](A,B,M)
\end{tikzpicture}
```

8.2. Barycentric coordinates

$pt_1, pt_2, \ldots, pt_n$ being n points, they define n vectors $\overrightarrow{v_1}, \overrightarrow{v_2}, \ldots, \overrightarrow{v_n}$ with the origin of the referential as the common endpoint. $\alpha_1, \alpha_2, \ldots\alpha_n$ are n numbers, the vector obtained by:

$$\frac{\alpha_1\overrightarrow{v_1} + \alpha_2\overrightarrow{v_2} + \cdots + \alpha_n\overrightarrow{v_n}}{\alpha_1 + \alpha_2 + \cdots + \alpha_n}$$

defines a single point.

\tkzDefBarycentricPoint(⟨pt1=α_1,pt2=α_2,...⟩)

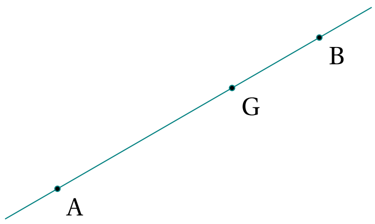
arguments	default	definition
(pt1=α_1,pt2=α_2,...)	no default	Each point has a assigned weight

You need at least two points. Result in `tkzPointResult`.

8.2.1. Using \tkzDefBarycentricPoint with two points

In the following example, we obtain the barycentre of points A and B with coefficients 1 and 2, in other words:

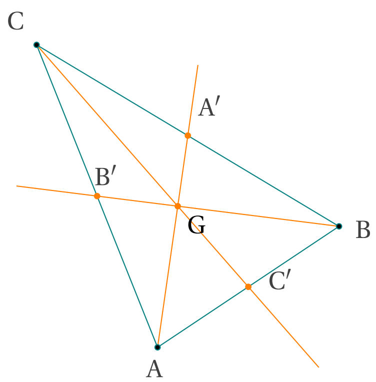
$$\overrightarrow{AI} = \frac{2}{3}\overrightarrow{AB}$$




```
\begin{tikzpicture}
  \tkzDefPoint(2,3){A}
  \tkzDefShiftPointCoord[2,3](30:4){B}
  \tkzDefBarycentricPoint(A=1,B=2)
  \tkzGetPoint{G}
  \tkzDrawLine(A,B)
  \tkzDrawPoints(A,B,G)
  \tkzLabelPoints(A,B,G)
\end{tikzpicture}
```

8.2.2. Using \tkzDefBarycentricPoint with three points

This time M is simply the center of gravity of the triangle.
For reasons of simplification and homogeneity, there is also `\tkzCentroid`.




```
\begin{tikzpicture}[scale=.8]
  \tkzDefPoints{2/1/A,5/3/B,0/6/C}
  \tkzDefBarycentricPoint(A=1,B=1,C=1)
  \tkzGetPoint{G}
  \tkzDefMidPoint(A,B)  \tkzGetPoint{C'}
  \tkzDefMidPoint(A,C)  \tkzGetPoint{B'}
  \tkzDefMidPoint(C,B)  \tkzGetPoint{A'}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawLines[add=0 and 1,new](A,G B,G C,G)
  \tkzLabelPoint(G){$G$}
  \tkzDrawPoints[new](A',B',C',G)
  \tkzDrawPoints(A,B,C)
  \tkzAutoLabelPoints[center=G](A,B,C)
  \tkzAutoLabelPoints[center=G,above right](A',B',C')
\end{tikzpicture}
```

8.3. Golden ration

From Wikipedia : In mathematics, two quantities are in the golden ratio if their ratio is the same as the ratio of their sum to the larger of the two quantities. Expressed algebraically, for quantities a, b a > b > 0 a + b is to a as a is to b.

$\frac{a+b}{a} = \frac{a}{b} = \phi = \frac{1+\sqrt{5}}{2}$

One of the two solutions to the equation $x^2 - x - 1 = 0$ is the golden ratio ϕ, $\phi = \frac{1+\sqrt{5}}{2}$.

`\tkzDefGoldenRatio(⟨pt1,pt2⟩)`

arguments	default	example
(pt1,pt2)	no default	\tkzDefGoldenRatio(A,C) \tkzGetPoint{B}

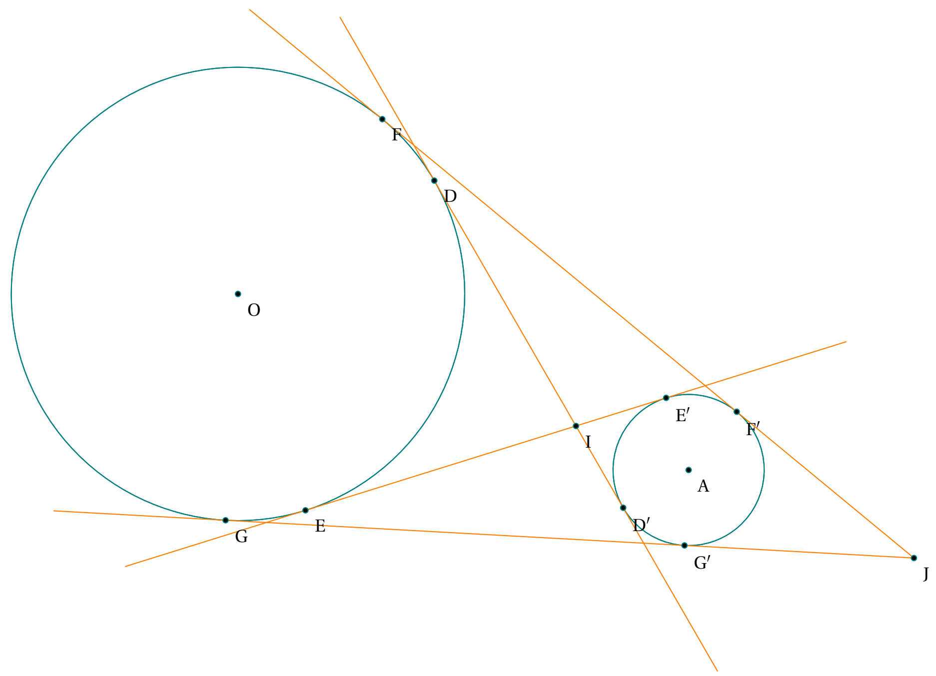
$AB = a$, $BC = b$ and $\frac{AC}{AB} = \frac{AB}{BC} = \phi$

8.4. Use the golden ratio to divide a line segment


```
\begin{tikzpicture}
 \tkzDefPoints{0/0/A,6/0/C}
 \tkzDefMidPoint(A,C) \tkzGetPoint{I}
 %\tkzDefPointWith[linear,K=\tkzInvPhi](A,C)
 \tkzDefGoldenRatio(A,C) \tkzGetPoint{B}
 \tkzDrawSegments(A,C)
 \tkzDrawPoints(A,B,C)
 \tkzLabelPoints(A,B,C)
\end{tikzpicture}
```

8.5. Internal Similitude Center

The centres of the two homotheties in which two circles correspond are called external and internal centres of similitude.



```
\begin{tikzpicture}[rotate=30]
 \tkzDefPoints{0/0/O,4/-5/A}
 \tkzDefPoints{3/0/x,5/-5/y}
 \pgfmathsetmacro\R{3}\pgfmathsetmacro\r{1}
 \tkzDefIntSimilitudeCenter[R](O,\R)(A,\r) \tkzGetPoint{I}
 \tkzDefExtSimilitudeCenter[R](O,\R)(A,\r) \tkzGetPoint{J}
 \tkzDefTangent[from  with R= I](O,3)   \tkzGetPoints{D}{E}
 \tkzDefTangent[from with R= I](A,1)    \tkzGetPoints{D'}{E'}
 \tkzDefTangent[from  with R= J](O,3)   \tkzGetPoints{F}{G}
 \tkzDefTangent[from with R= J](A,1)    \tkzGetPoints{F'}{G'}
 \tkzDrawCircles(O,x A,y)               \tkzDrawCircles[R](O,3 A,1)
 \tkzDrawSegments[add = .5 and .5,new](D,D' E,E')
 \tkzDrawSegments[add= 0 and 0.25,new](J,F J,G)
 \tkzDrawPoints(O,A,I,J,D,E,F,G,D',E',F',G')
 \tkzLabelPoints[font=\scriptsize](O,A,I,J,D,E,F,G,D',E',F',G')
\end{tikzpicture}
```

You can **`\tkzDefBarycentricPoint`** to find a homothetic center

```
\tkzDefBarycentricPoint(O=\r,A=\R)      \tkzGetPoint{I}
\tkzDefBarycentricPoint(O={-\r},A=\R)   \tkzGetPoint{J}
```

8.6. Equidistant points

8.6.1. \tkzDefEquiPoints

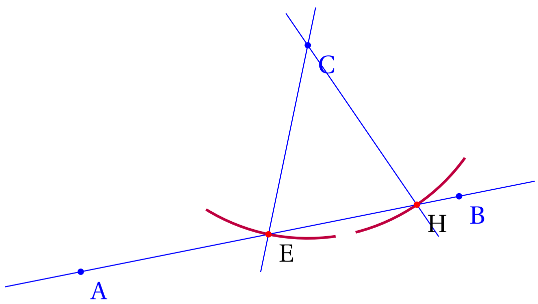
`\tkzDefEquiPoints[⟨local options⟩](⟨pt1,pt2⟩)`

arguments	default	definition
(pt1,pt2)	no default	unordered list of two items

options	default	definition
dist	2 (cm)	half the distance between the two points
from=pt	no default	reference point
show	false	if true displays compass traces
/compass/delta	0	compass trace size

This macro makes it possible to obtain two points on a straight line equidistant from a given point.

8.6.2. Using \tkzDefEquiPoints with options




```
\begin{tikzpicture}
  \tkzSetUpCompass[color=purple,line width=1pt]
  \tkzDefPoints{0/1/A,5/2/B,3/4/C}
  \tkzDefEquiPoints[from=C,dist=1,show,
      /tkzcompass/delta=20](A,B)
   \tkzGetPoints{E}{H}
   \tkzDrawLines[color=blue](C,E C,H A,B)
   \tkzDrawPoints[color=blue](A,B,C)
   \tkzDrawPoints[color=red](E,H)
   \tkzLabelPoints(E,H)
   \tkzLabelPoints[color=blue](A,B,C)
\end{tikzpicture}
```

9. Special points relating to a triangle

9.1. Triangle center: \tkzDefTriangleCenter

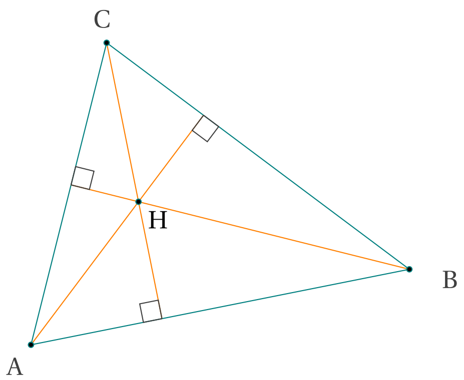
```
\tkzDefTriangleCenter[⟨local options⟩](⟨A,B,C⟩)
```

This macro allows you to define the center of a triangle.. Be careful, the arguments are lists of three points. This macro is used in conjunction with `\tkzGetPoint` to get the center you are looking for.
You can use `tkzPointResult` if it is not necessary to keep the results.

arguments	default	example
(pt1,pt2,pt3)	no default	\tkzDefTriangleCenter[ortho](B,C,A)
options	**default**	**definition**
ortho	circum	intersection of the altitudes
orthic	circum	…
centroid	circum	intersection of the medians
median	circum	…
circum	circum	circle center circumscribed
in	circum	center of the circle inscribed in a triangle
in	circum	intersection of the bisectors
ex	circum	center of a circle exinscribed to a triangle
euler	circum	center of Euler's circle
gergonne	circum	defined with the Contact triangle
symmedian	circum	Lemoine's point or symmedian center or Grebe's point
lemoine	circum	…
grebe	circum	…
spieker	circum	Spieker circle center
nagel	circum	Nagel Center
mittenpunkt	circum	Or middlespoint
feuerbach	circum	Feuerbach Point

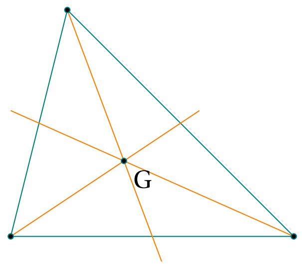
9.1.1. Option ortho or orthic

The intersection H of the three altitudes of a triangle is called the orthocenter.



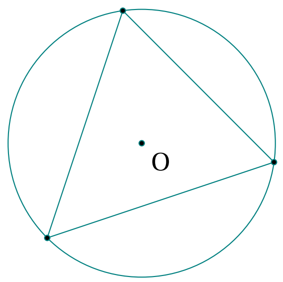

```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(5,1){B}
  \tkzDefPoint(1,4){C}
  \tkzDefTriangleCenter[ortho](B,C,A)
  \tkzGetPoint{H}
  \tkzDefSpcTriangle[orthic,name=H](A,B,C){a,b,c}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawSegments[new](A,Ha B,Hb C,Hc)
  \tkzDrawPoints(A,B,C,H)
  \tkzLabelPoint(H){$H$}
  \tkzAutoLabelPoints[center=H](A,B,C)
 \tkzMarkRightAngles(A,Ha,B B,Hb,C C,Hc,A)
\end{tikzpicture}
```

9.1.2. Option centroid




```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoints{0/0/A,5/0/B,1/4/C}
  \tkzDefTriangleCenter[centroid](A,B,C)
  \tkzGetPoint{G}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawLines[add = 0 and 2/3,new](A,G B,G C,G)
  \tkzDrawPoints(A,B,C,G)
  \tkzLabelPoint(G){$G$}
\end{tikzpicture}
```

9.1.3. Option circum

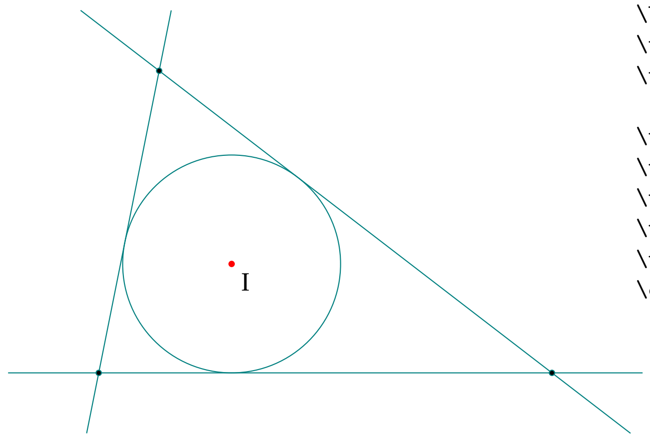



```
 \begin{tikzpicture}
  \tkzDefPoints{0/1/A,3/2/B,1/4/C}
  \tkzDefTriangleCenter[circum](A,B,C)
  \tkzGetPoint{O}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawCircle(O,A)
  \tkzDrawPoints(A,B,C,O)
  \tkzLabelPoint(O){$O$}
\end{tikzpicture}
```

9.1.4. Option in

In geometry, the incircle or inscribed circle of a triangle is the largest circle contained in the triangle; it touches (is tangent to) the three sides. The center of the incircle is a triangle center called the triangle's incenter. The center of the incircle, called the incenter, can be found as the intersection of the three internal angle bisectors. The center of an excircle is the intersection of the internal bisector of one angle (at vertex A, for example) and the external bisectors of the other two. The center of this excircle is called the excenter relative to the vertex A, or the excenter of A. Because the internal bisector of an angle is perpendicular to its external bisector, it follows that the center of the incircle together with the three excircle centers form an orthocentric system.
(`https://en.wikipedia.org/wiki/Incircle_and_excircles_of_a_triangle`)

We get the centre of the inscribed circle of the triangle. The result is of course in **tkzPointResult**. We can retrieve it with `\tkzGetPoint`.

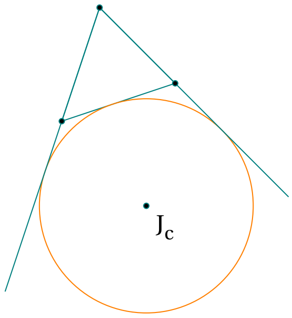



```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,6/0/B,0.8/4/C}
\tkzDefTriangleCenter[in](A,B,C)
   \tkzGetPoint{I}
\tkzDrawLines(A,B B,C C,A)
\tkzDrawCircle[in](A,B,C)
\tkzDrawPoint[red](I)
\tkzDrawPoints(A,B,C)
\tkzLabelPoint(I){$I$}
\end{tikzpicture}
```

9.1.5. Option ex

An excircle or escribed circle of the triangle is a circle lying outside the triangle, tangent to one of its sides and tangent to the extensions of the other two. Every triangle has three distinct excircles, each tangent to one of the

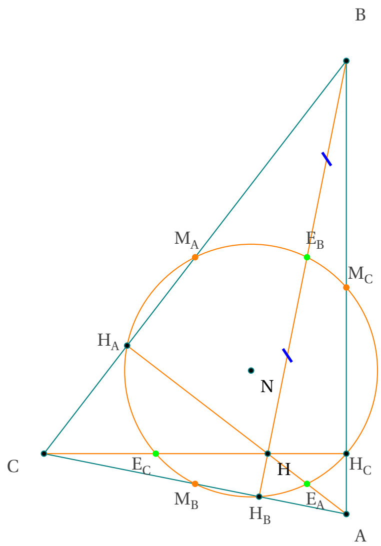
triangle's sides.
(https://en.wikipedia.org/wiki/Incircle_and_excircles_of_a_triangle)
We get the centre of an inscribed circle of the triangle. The result is of course in **tkzPointResult**. We can retrieve it with `\tkzGetPoint`.




```
\begin{tikzpicture}[scale=.5]
  \tkzDefPoints{0/1/A,3/2/B,1/4/C}
  \tkzDefTriangleCenter[ex](B,C,A)
  \tkzGetPoint{J_c}
  \tkzDefPointBy[projection=onto A--B](J_c)
  \tkzGetPoint{Tc}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawCircle[new](J_c,Tc)
  \tkzDrawLines[add=1.5 and 0](A,C B,C)
  \tkzDrawPoints(A,B,C,J_c)
  \tkzLabelPoints(J_c)
\end{tikzpicture}
```

9.1.6. Option euler

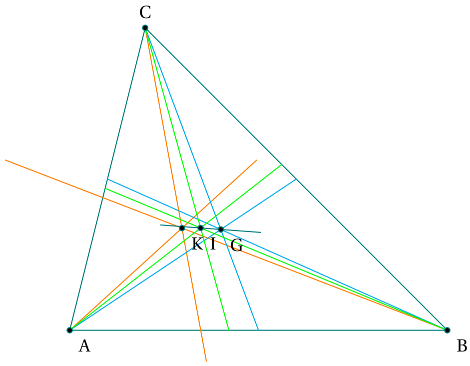
This macro allows to obtain the center of the circle of the nine points or euler's circle or Feuerbach's circle. The nine-point circle, also called Euler's circle or the Feuerbach circle, is the circle that passes through the perpendicular feet H_A, H_B, and H_C dropped from the vertices of any reference triangle ABC on the sides opposite them. Euler showed in 1765 that it also passes through the midpoints M_A, M_B, M_C of the sides of ABC. By Feuerbach's theorem, the nine-point circle also passes through the midpoints E_A, E_B, and E_C of the segments that join the vertices and the orthocenter H. These points are commonly referred to as the Euler points.
(https://mathworld.wolfram.com/Nine-PointCircle.html)




```
\begin{tikzpicture}[scale=1,rotate=90]
 \tkzDefPoints{0/0/A,6/0/B,0.8/4/C}
 \tkzDefSpcTriangle[medial,name=M](A,B,C){_A,_B,_C}
 \tkzDefTriangleCenter[euler](A,B,C)\tkzGetPoint{N}
 % I= N nine points
 \tkzDefTriangleCenter[ortho](A,B,C)\tkzGetPoint{H}
 \tkzDefMidPoint(A,H) \tkzGetPoint{E_A}
 \tkzDefMidPoint(C,H) \tkzGetPoint{E_C}
 \tkzDefMidPoint(B,H) \tkzGetPoint{E_B}
 \tkzDefSpcTriangle[ortho,name=H](A,B,C){_A,_B,_C}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawCircle[new](N,E_A)
 \tkzDrawSegments[new](A,H_A B,H_B C,H_C)
 \tkzDrawPoints(A,B,C,N,H)
 \tkzDrawPoints[new](M_A,M_B,M_C)
 \tkzDrawPoints( H_A,H_B,H_C)
 \tkzDrawPoints[green](E_A,E_B,E_C)
 \tkzAutoLabelPoints[center=N,
 font=\scriptsize](A,B,C,M_A,M_B,M_C,H_A,H_B,H_C,E_A,E_B,E_C)
 \tkzLabelPoints[font=\scriptsize](H,N)
 \tkzMarkSegments[mark=s|,size=3pt,
 color=blue,line width=1pt](B,E_B E_B,H)
\end{tikzpicture}
```

9.1.7. Option symmedian

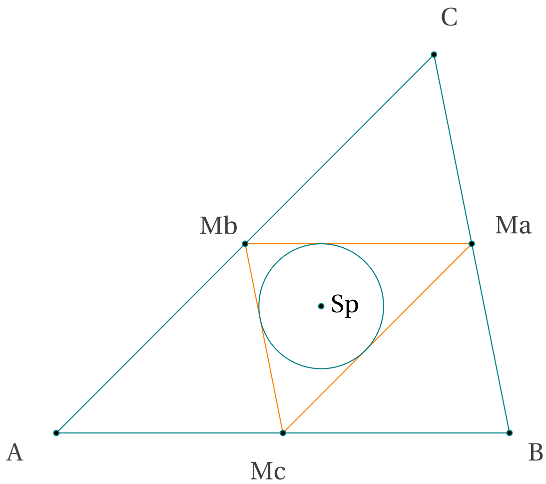
The point of concurrence K of the symmedians, sometimes also called the Lemoine point (in England and France) or the Grebe point (in Germany).
Weisstein, Eric W. "Symmedian Point." From MathWorld–A Wolfram Web Resource.




```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(5,0){B}
  \tkzDefPoint(1,4){C}
  \tkzDefTriangleCenter[symmedian](A,B,C)\tkzGetPoint{K}
  \tkzDefTriangleCenter[median](A,B,C)\tkzGetPoint{G}
  \tkzDefTriangleCenter[in](A,B,C)\tkzGetPoint{I}
  \tkzDefSpcTriangle[centroid,name=M](A,B,C){a,b,c}
  \tkzDefSpcTriangle[incentral,name=I](A,B,C){a,b,c}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawLines[add = 0 and 2/3,new](A,K B,K C,K)
  \tkzDrawSegments[color=cyan](A,Ma B,Mb C,Mc)
  \tkzDrawSegments[color=green](A,Ia B,Ib C,Ic)
  \tkzDrawLine[add=2 and 2](G,I)
  \tkzDrawPoints(A,B,C,K,G,I)
  \tkzLabelPoints[font=\scriptsize](A,B,K,G,I)
  \tkzLabelPoints[above,font=\scriptsize](C)
\end{tikzpicture}
```

9.1.8. Option spieker

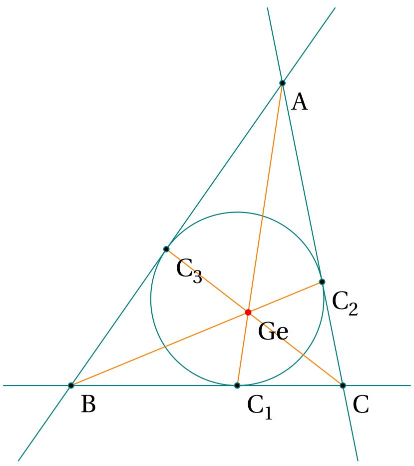
The Spieker center is the center Sp of the Spieker circle, i.e., the incenter of the medial triangle of a reference triangle.
Weisstein, Eric W. "Spieker Center." From MathWorld–A Wolfram Web Resource.




```
\begin{tikzpicture}
 \tkzDefPoints{0/0/A,6/0/B,5/5/C}
 \tkzDefSpcTriangle[medial](A,B,C){Ma,Mb,Mc}
 \tkzDefTriangleCenter[centroid](A,B,C)
 \tkzGetPoint{G}
 \tkzDefTriangleCenter[spieker](A,B,C)
 \tkzGetPoint{Sp}
 \tkzDrawPolygon[](A,B,C)
 \tkzDrawPolygon[new](Ma,Mb,Mc)
 \tkzDrawCircle[in](Ma,Mb,Mc)
 \tkzDrawPoints(B,C,A,Sp,Ma,Mb,Mc)
 \tkzAutoLabelPoints[center=G,dist=.3](Ma,Mb,Mc)
 \tkzLabelPoints[right](Sp)
 \tkzAutoLabelPoints[center=G](A,B,C)
\end{tikzpicture}
```

9.1.9. Option gergonne

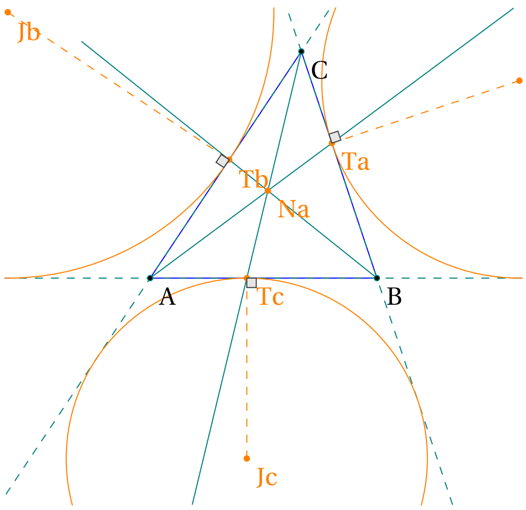
The Gergonne Point is the point of concurrency which results from connecting the vertices of a triangle to the opposite points of tangency of the triangle's incircle. (Joseph Gergonne French mathematician)




```
\begin{tikzpicture}
\tkzDefPoints{0/0/B,3.6/0/C,2.8/4/A}
\tkzDefTriangleCenter[gergonne](A,B,C)
\tkzGetPoint{Ge}
\tkzDefSpcTriangle[intouch](A,B,C){C_1,C_2,C_3}
\tkzDrawCircle[in](A,B,C)
\tkzDrawLines[add=.25 and .25,teal](A,B A,C B,C)
\tkzDrawSegments[new](A,C_1 B,C_2 C,C_3)
\tkzDrawPoints(A,...,C,C_1,C_2,C_3)
\tkzDrawPoints[red](Ge)
\tkzLabelPoints(A,...,C,C_1,C_2,C_3,Ge)
\end{tikzpicture}
```

9.1.10. Option `nagel`

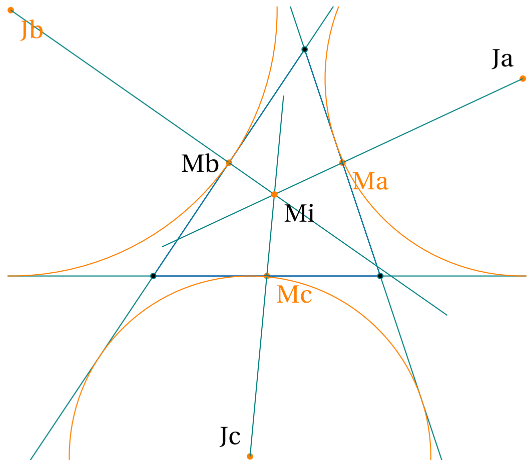
Let Ta be the point at which the excircle with center Ja meets the side BC of a triangle ABC, and define Tb and Tc similarly. Then the lines ATa, BTb, and CTc concur in the Nagel point Na.
Weisstein, Eric W. "Nagel point." From MathWorld–A Wolfram Web Resource.




```
\begin{tikzpicture}[scale=.5]
\tkzDefPoints{0/0/A,6/0/B,4/6/C}
\tkzDefSpcTriangle[ex](A,B,C){Ja,Jb,Jc}
\tkzDefSpcTriangle[extouch](A,B,C){Ta,Tb,Tc}
\tkzDefTriangleCenter[nagel](A,B,C)
\tkzGetPoint{Na}
\tkzDrawPolygon[blue](A,B,C)
\tkzDrawLines[add=0 and 1](A,Ta B,Tb C,Tc)
\tkzDrawPoints[new](Ja,Jb,Jc,Ta,Tb,Tc)
\tkzClipBB
\tkzDrawLines[add=1 and 1,dashed](A,B B,C C,A)
\tkzDrawCircles[ex,new](A,B,C C,A,B B,C,A)
\tkzDrawSegments[new,dashed](Ja,Ta Jb,Tb Jc,Tc)
\tkzDrawPoints(B,C,A)
\tkzDrawPoints[new](Na)
\tkzLabelPoints(B,C,A)
\tkzLabelPoints[new](Na)
\tkzLabelPoints[new](Ja,Jb,Jc,Ta,Tb,Tc)
\tkzMarkRightAngles[fill=gray!20](Ja,Ta,C
            Jb,Tb,A Jc,Tc,B)
\end{tikzpicture}
```

9.1.11. Option `mittenpunkt`

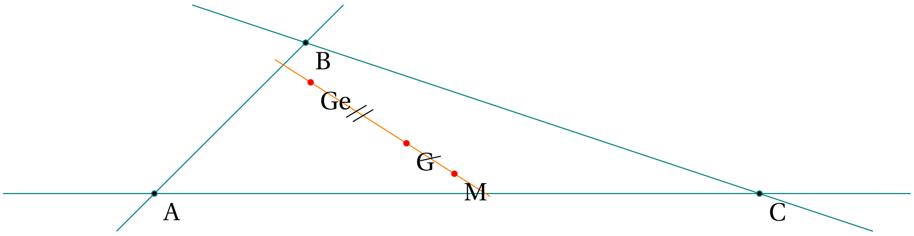
The mittenpunkt (also called the middlespoint) of a triangle ABC is the symmedian point of the excentral triangle, i.e., the point of concurrence M of the lines from the excenters through the corresponding triangle side midpoints.
Weisstein, Eric W. "Mittenpunkt." From MathWorld–A Wolfram Web Resource.




```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoints{0/0/A,6/0/B,4/6/C}
 \tkzDefSpcTriangle[centroid](A,B,C){Ma,Mb,Mc}
 \tkzDefSpcTriangle[ex](A,B,C){Ja,Jb,Jc}
 \tkzDefSpcTriangle[extouch](A,B,C){Ta,Tb,Tc}
 \tkzDefTriangleCenter[mittenpunkt](A,B,C)
 \tkzGetPoint{Mi}
 \tkzDrawPoints[new](Ma,Mb,Mc,Ja,Jb,Jc)
 \tkzClipBB
 \tkzDrawPolygon[blue](A,B,C)
 \tkzDrawLines[add=0 and 1](Ja,Ma
              Jb,Mb Jc,Mc)
 \tkzDrawLines[add=1 and 1](A,B A,C B,C)
 \tkzDrawCircles[new](Ja,Ta Jb,Tb Jc,Tc)
 \tkzDrawPoints(B,C,A)
 \tkzDrawPoints[new](Mi)
 \tkzLabelPoints(Mi)
 \tkzLabelPoints[left](Mb)
 \tkzLabelPoints[new](Ma,Mc,Jb,Jc)
 \tkzLabelPoints[above left](Ja,Jc)
\end{tikzpicture}
```

9.1.12. Example : relation between gergonne, centroid and mittenpunkt

The Gergonne point Ge, triangle centroid G, and mittenpunkt M are collinear, with GeG/GM=2.




```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,2/2/B,8/0/C}
\tkzDefTriangleCenter[gergonne](A,B,C) \tkzGetPoint{Ge}
\tkzDefTriangleCenter[centroid](A,B,C)
\tkzGetPoint{G}
\tkzDefTriangleCenter[mittenpunkt](A,B,C)
\tkzGetPoint{M}
\tkzDrawLines[add=.25 and .25,teal](A,B A,C B,C)
\tkzDrawLines[add=.25 and .25,new](Ge,M)
\tkzDrawPoints(A,...,C)
\tkzDrawPoints[red,size=2](G,M,Ge)
\tkzLabelPoints(A,...,C,M,G,Ge)
\tkzMarkSegment[mark=s||](Ge,G)
\tkzMarkSegment[mark=s|](G,M)
\end{tikzpicture}
```

10. Projection of excenters

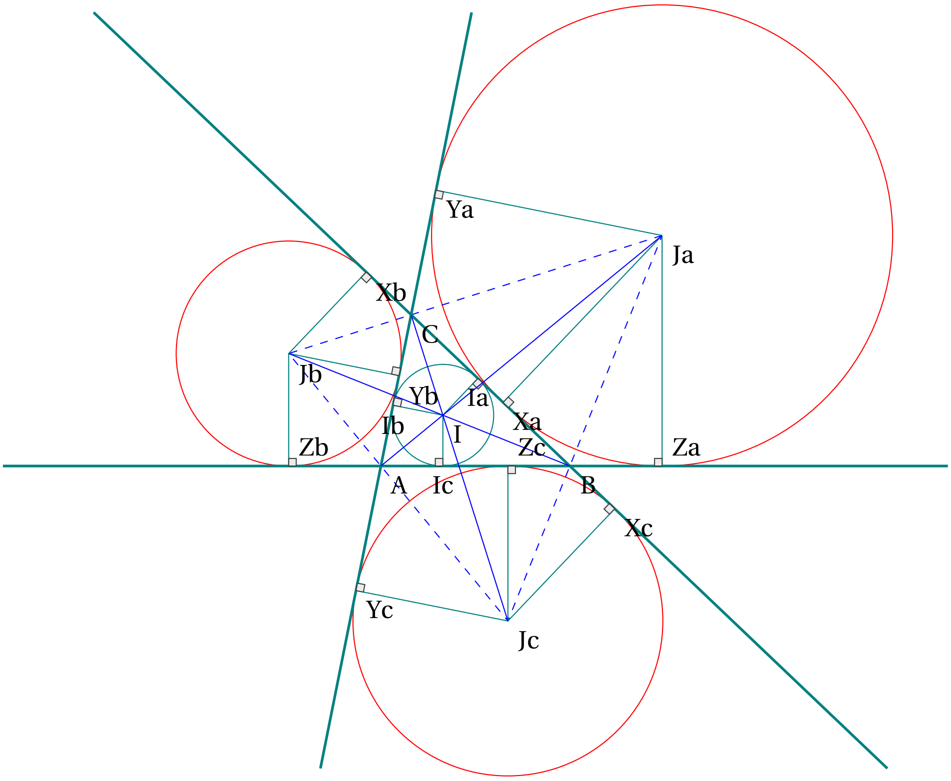
`\tkzDefProjExcenter[⟨local options⟩](⟨A,B,C⟩)(⟨a,b,c⟩){⟨X,Y,Z⟩}`

Each excenter has three projections on the sides of the triangle ABC. We can do this with one macro `\tkzDefProjExcenter[name=J](A,B,C)(a,b,c){Y,Z,X}`.

options	default	definition
name	no defaut	used to name the vertices

arguments	default	definition
(pt1=α_1,pt2=α_2,...)	no default	Each point has a assigned weight

10.0.1. Excircles




```
\begin{tikzpicture}[scale=.5]
\tkzDefPoints{0/0/A,5/0/B,0.8/4/C}
\tkzDefSpcTriangle[excentral,name=J](A,B,C){a,b,c}
\tkzDefSpcTriangle[intouch,name=I](A,B,C){a,b,c}
\tkzDefProjExcenter[name=J](A,B,C)(a,b,c){X,Y,Z}

\tkzDefCircle[in](A,B,C)   \tkzGetPoint{I} \tkzGetSecondPoint{T}
\tkzDrawCircles[red](Ja,Xa Jb,Yb Jc,Zc)
\tkzDrawCircle(I,T)
\tkzDrawPolygon[dashed,color=blue](Ja,Jb,Jc)
\tkzDrawLines[add=2 and 2,line width=1pt](A,C A,B B,C)
\tkzDrawSegments(Ja,Xa Ja,Ya Ja,Za
```

```
                  Jb,Xb Jb,Yb Jb,Zb
                  Jc,Xc Jc,Yc Jc,Zc
                  I,Ia I,Ib I,Ic)
\tkzMarkRightAngles[size=.2,fill=gray!15](%
      Ja,Za,B
      Ja,Xa,B
      Ja,Ya,C
      Jb,Yb,C
      Jb,Zb,B
      Jb,Xb,C
      Jc,Yc,A
      Jc,Zc,B
      Jc,Xc,C
      I,Ia,B
      I,Ib,C
      I,Ic,A)
\tkzDrawSegments[blue](Jc,C Ja,A Jb,B)
\tkzLabelPoints(Xb,Yc,A,B,C,Xa,Xc,Ya,Yb,Ja,Jb,Jc,I)
\tkzLabelPoints[above right](Za,Zb,Zc)
\tkzLabelPoints[below](Ia,Ib,Ic)
\end{tikzpicture}
```

11. Point on line or circle

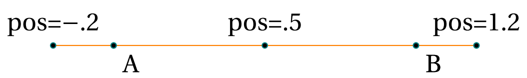
11.1. Point on a line

\tkzDefPointOnLine[⟨local options⟩](⟨A,B⟩)

arguments	default	definition
pt1,pt2	no default	Two points to define a line

options	default	definition
pos=nb		nb is a decimal

11.1.1. Use of option pos

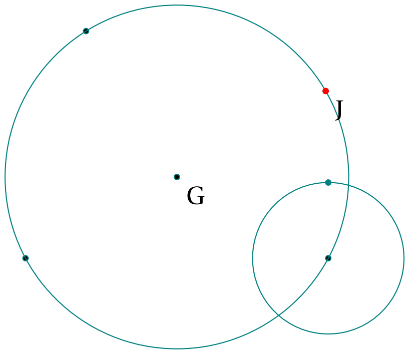



```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,4/0/B}
\tkzDefPointOnLine[pos=1.2](A,B)
\tkzGetPoint{P}
\tkzDefPointOnLine[pos=-0.2](A,B)
\tkzGetPoint{R}
\tkzDefPointOnLine[pos=0.5](A,B)
\tkzGetPoint{S}
\tkzDrawLine[new](A,B)
\tkzDrawPoints(A,B,P)
\tkzLabelPoints(A,B)
\tkzLabelPoint[above](P){pos=$1.2$}
\tkzLabelPoint[above](R){pos=$-.2$}
\tkzLabelPoint[above](S){pos=$.5$}
\tkzDrawPoints(A,B,P,R,S)
\tkzLabelPoints(A,B)
\end{tikzpicture}
```

11.2. Point on a circle

\tkzDefPointOnCircle[⟨local options⟩]

options	default	definition
angle	0	angle formed with the abscissa axis
center	tkzPointResult	circle center required
radius	\tkzLengthResult	radius circle




```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,4/0/B,0.8/3/C}
\tkzDefPointOnCircle[angle=90,center=B,radius=1]
\tkzGetPoint{I}
\tkzDefCircle[circum](A,B,C)
\tkzGetPoint{G} \tkzGetLength{rG}
\tkzDefPointOnCircle[angle=30,center=G,radius=\rG]
\tkzGetPoint{J}
\tkzDrawCircle[R,teal](B,1)
\tkzDrawPoint[teal](I)
\tkzDrawPoints(A,B,C)
\tkzDrawCircle(G,J)
\tkzDrawPoints(G,J)
\tkzDrawPoint[red](J)
\tkzLabelPoints(G,J)
\end{tikzpicture}
```

12. Definition of points by transformation : \tkzDefPointBy

These transformations are:

– translation;

– homothety;

– orthogonal reflection or symmetry;

– central symmetry;

– orthogonal projection;

– rotation (degrees or radians);

– inversion with respect to a circle.

The choice of transformations is made through the options. There are two macros, one for the transformation of a single point `\tkzDefPointBy` and the other for the transformation of a list of points `\tkzDefPointsBy`. By default the image of A is A′. For example, we'll write:

```
\tkzDefPointBy[translation= from A to A'](B)
```

The result is in `tkzPointResult`

`\tkzDefPointBy[⟨local options⟩](⟨pt⟩)`

The argument is a simple existing point and its image is stored in `tkzPointResult`. If you want to keep this point then the macro `\tkzGetPoint{M}` allows you to assign the name `M` to the point.

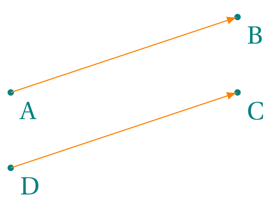
arguments	definition	examples
pt	existing point name	(A)

options		examples
translation	= from #1 to #2	[translation=from A to B](E)
homothety	= center #1 ratio #2	[homothety=center A ratio .5](E)
reflection	= over #1--#2	[reflection=over A--B](E)
symmetry	= center #1	[symmetry=center A](E)
projection	= onto #1--#2	[projection=onto A--B](E)
rotation	= center #1 angle #2	[rotation=center O angle 30](E)
rotation in rad	= center #1 angle #2	[rotation in rad=center O angle pi/3](E)
inversion	= center #1 through #2	[inversion =center O through A](E)
inversion negative	= center #1 through #2	...

The image is only defined and not drawn.

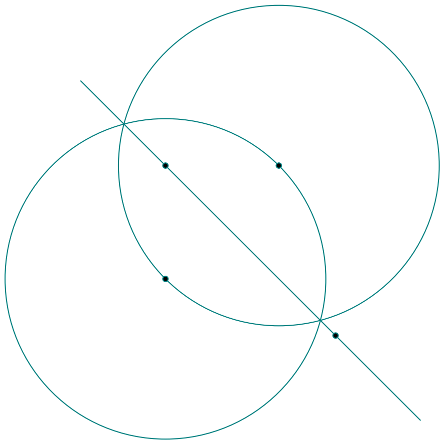
12.1. Examples of transformations

12.1.1. translation



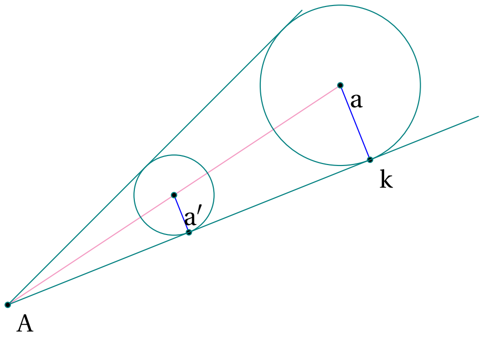

```
\begin{tikzpicture}[>=latex]
 \tkzDefPoints{0/0/A,3/1/B,3/0/C}
 \tkzDefPointBy[translation= from B to A](C)
 \tkzGetPoint{D}
 \tkzDrawPoints[teal](A,B,C,D)
 \tkzLabelPoints[color=teal](A,B,C,D)
 \tkzDrawSegments[orange,->](A,B D,C)
\end{tikzpicture}
```

12.1.2. reflection (orthogonal symmetry)



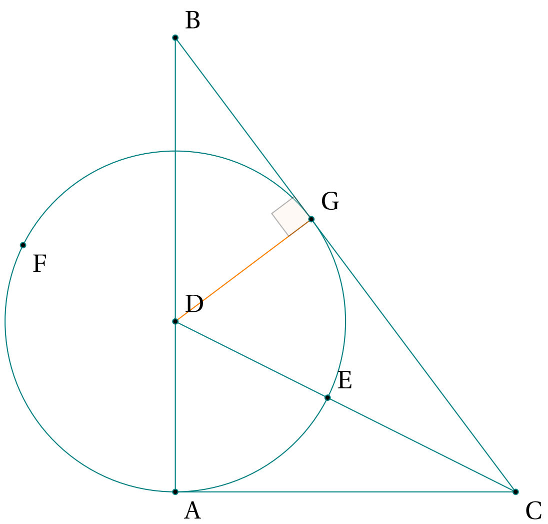
```
\begin{tikzpicture}[scale=.75]
 \tkzDefPoints{-2/-2/A,-1/-1/C,-4/2/D,-4/0/O}
 \tkzDrawCircle(O,A)
 \tkzDefPointBy[reflection = over C--D](A)
 \tkzGetPoint{A'}
 \tkzDefPointBy[reflection = over C--D](O)
 \tkzGetPoint{O'}
 \tkzDrawCircle(O',A')
 \tkzDrawLine[add= .5 and .5](C,D)
 \tkzDrawPoints(C,D,O,O')
\end{tikzpicture}
```

12.1.3. homothety and projection



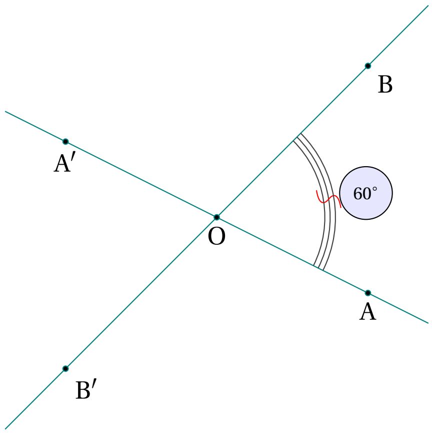

```
\begin{tikzpicture}
  \tkzDefPoints{0/1/A,5/3/B,3/4/C}
  \tkzDefLine[bisector](B,A,C) \tkzGetPoint{a}
  \tkzDrawLine[add=0 and 0,color=magenta!50 ](A,a)
  \tkzDefPointBy[homothety=center A ratio .5](a)
  \tkzGetPoint{a'}
  \tkzDefPointBy[projection = onto A--B](a')
  \tkzGetPoint{k'}
  \tkzDefPointBy[projection = onto A--B](a)
  \tkzGetPoint{k}
  \tkzDrawLines[add= 0 and .3](A,k A,C)
  \tkzDrawSegments[blue](a',k' a,k)
  \tkzDrawPoints(a,a',k,k',A)
  \tkzDrawCircles(a',k' a,k)
  \tkzLabelPoints(a,a',k,A)
\end{tikzpicture}
```

12.1.4. projection



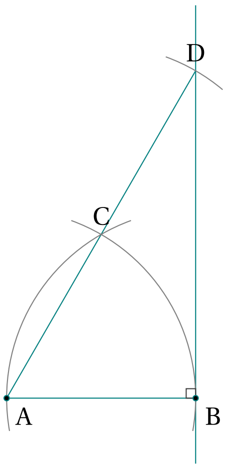

```
\begin{tikzpicture}[scale=1.5]
 \tkzDefPoints{0/0/A,0/4/B}
 \tkzDefTriangle[pythagore](B,A) \tkzGetPoint{C}
 \tkzDefLine[bisector](B,C,A) \tkzGetPoint{c}
 \tkzInterLL(C,c)(A,B)  \tkzGetPoint{D}
 \tkzDefPointBy[projection=onto B--C](D)
 \tkzGetPoint{G}
 \tkzInterLC(C,D)(D,A) \tkzGetPoints{E}{F}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawSegment(C,D)
 \tkzDrawCircle(D,A)
 \tkzDrawSegment[new](D,G)
 \tkzMarkRightAngle[fill=orange!10,opacity=.4](D,G,B)
 \tkzDrawPoints(A,C,F) \tkzLabelPoints(A,C,F)
 \tkzDrawPoints(B,D,E,G)
 \tkzLabelPoints[above right](B,D,E,G)
\end{tikzpicture}
```

12.1.5. symmetry



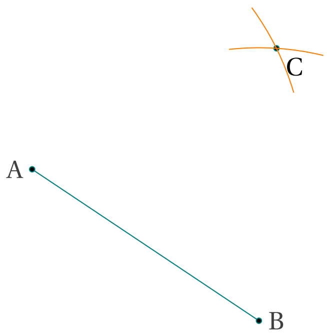

```
\begin{tikzpicture}[scale=1]
  \tkzDefPoints{2/-1/A,2/2/B,0/0/O}
  \tkzDefPointsBy[symmetry=center O](B,A){}
  \tkzDrawLine(A,A')
  \tkzDrawLine(B,B')
  \tkzMarkAngle[mark=s,arc=lll,
      size=1.5,mkcolor=red](A,O,B)
  \tkzLabelAngle[pos=2,circle,draw,
    fill=blue!10,font=\scriptsize](A,O,B){$60^{\circ}$}
  \tkzDrawPoints(A,B,O,A',B')
  \tkzLabelPoints(B,B')
  \tkzLabelPoints[below](A,O,A')
\end{tikzpicture}
```

12.1.6. rotation




```
\begin{tikzpicture}[scale=0.5]
 \tkzDefPoints{0/0/A,5/0/B}
 \tkzDrawSegment(A,B)
 \tkzDefPointBy[rotation=center A angle 60](B)
 \tkzGetPoint{C}
 \tkzDefPointBy[symmetry=center C](A)
 \tkzGetPoint{D}
 \tkzDrawSegment(A,tkzPointResult)
 \tkzDrawLine(B,D)
 \tkzDrawArc(A,B)(C)
 \tkzDrawArc(B,C)(A)
 \tkzDrawArc(C,D)(D)
 \tkzMarkRightAngle(D,B,A)
 \tkzDrawPoints(A,B)
 \tkzLabelPoints(A,B)
 \tkzLabelPoints[above](C,D)
\end{tikzpicture}
```

12.1.7. rotation in radian




```
\begin{tikzpicture}
  \tkzDefPoint["$A$" left](1,5){A}
  \tkzDefPoint["$B$" right](4,3){B}
  \tkzDefPointBy[rotation in rad= center A angle pi/3](B)
  \tkzGetPoint{C}
  \tkzDrawSegment(A,B)
  \tkzDrawPoints(A,B,C)
  \tkzCompass(A,C)
  \tkzCompass(B,C)
  \tkzLabelPoints(C)
\end{tikzpicture}
```

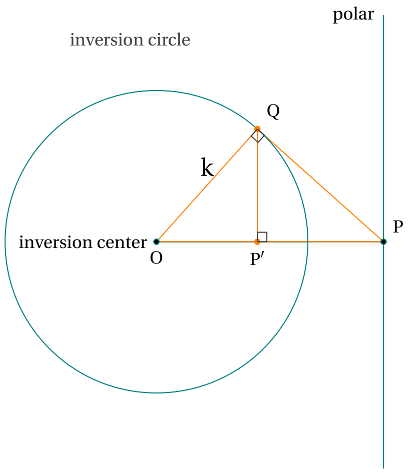
12.1.8. inversion

Inversion is the process of transforming points to a corresponding set of points known as their inverse points. Two points P and P′ are said to be inverses with respect to an inversion circle having inversion center O and inversion radius k if P′ is the perpendicular foot of the altitude of OQP, where Q is a point on the circle such that

OQ is perpendicular to PQ.
The quantity k^2 is known as the circle power (Coxeter 1969, p. 81). (https://mathworld.wolfram.com/Inversion.html)
Some propositions :

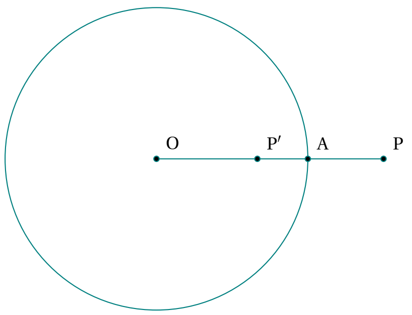
- The inverse of a circle (not through the center of inversion) is a circle.
- The inverse of a circle through the center of inversion is a line.
- The inverse of a line (not through the center of inversion) is a circle through the center of inversion.
- A circle orthogonal to the circle of inversion is its own inverse.
- A line through the center of inversion is its own inverse.
- Angles are preserved in inversion.

Explanation



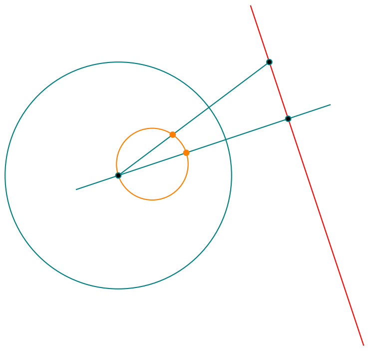

```
\begin{tikzpicture}[scale=.5]
  \tkzDefPoints{4/0/A,6/0/P,0/0/O}
  \tkzDefCircle(O,A)
  \tkzDefLine[orthogonal=through P](O,P)
  \tkzGetPoint{L}
  \tkzDefTangent[from = P](O,A) \tkzGetPoints{Q}{R}
  \tkzDefPointBy[projection=onto O--A](Q) \tkzGetPoint{P'}
  \tkzDrawSegments(O,P O,A)
  \tkzDrawSegments[new](O,P O,Q P,Q Q,P')
  \tkzDrawCircle(O,A)
  \tkzDrawLines[add=1 and 0](P,L)
  \tkzLabelPoints[below,font=\scriptsize](O,P')
  \tkzLabelPoints[above right,font=\scriptsize](P,Q)
  \tkzDrawPoints(O,P) \tkzDrawPoints[new](Q,P')
  \tkzLabelSegment[above](O,Q){$k$}
  \tkzMarkRightAngles(A,P',Q P,Q,O)
  \tkzLabelCircle[above=.5cm,
      font=\scriptsize](O,A)(100){inversion circle}
  \tkzLabelPoint[left,font=\scriptsize](O){inversion center}
  \tkzLabelPoint[left,font=\scriptsize](L){polar}
\end{tikzpicture}
```

Directly (Center O power=$k^2 = OA^2 = OP \times OP'$)

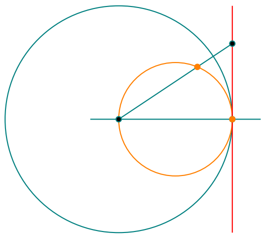



```
\begin{tikzpicture}[scale=.5]
  \tkzDefPoints{4/0/A,6/0/P,0/0/O}
  \tkzDefCircle(O,A)
  \tkzDefPointBy[inversion = center O through A](P)
  \tkzGetPoint{P'}
  \tkzDrawSegments(O,P)
  \tkzDrawCircle(O,A)
  \tkzLabelPoints[above right,font=\scriptsize](O,A,P,P')
  \tkzDrawPoints(O,A,P,P')
\end{tikzpicture}
```

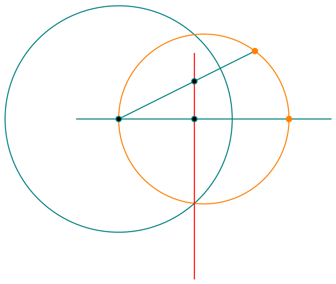
12.1.9. Inversion of lines



```
\begin{tikzpicture}[scale=.5]
\tkzDefPoints{0/0/O,3/0/I,4/3/P,6/-3/Q}
\tkzDrawCircle(O,I)
\tkzDefPointBy[projection= onto P--Q](O) \tkzGetPoint{A}
\tkzDefPointBy[inversion = center O through I](A)
\tkzGetPoint{A'}
\tkzDefPointBy[inversion = center O through I](P)
\tkzGetPoint{P'}
\tkzDrawCircle[new,diameter](O,A')
\tkzDrawLines[add=.25 and .25,red](P,Q)
\tkzDrawLines[add=.25 and .25](O,A)
\tkzDrawSegments(O,P)
\tkzDrawPoints(A,P,O) \tkzDrawPoints[new](A',P')
\end{tikzpicture}
```

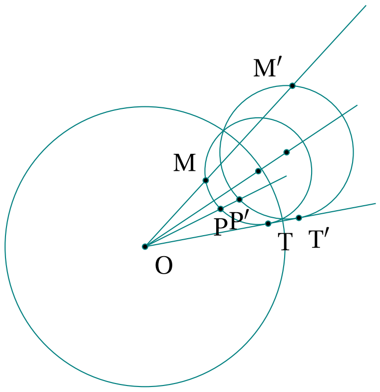


```
\begin{tikzpicture}[scale=.5]
\tkzDefPoints{0/0/O,3/0/I,3/2/P,3/-2/Q}
\tkzDrawCircle(O,I)
\tkzDefPointBy[projection= onto P--Q](O) \tkzGetPoint{A}
\tkzDefPointBy[inversion = center O through I](A)
\tkzGetPoint{A'}
\tkzDefPointBy[inversion = center O through I](P)
\tkzGetPoint{P'}
\tkzDrawCircle[new,diameter](O,A')
\tkzDrawLines[add=.25 and .25,red](P,Q)
\tkzDrawLines[add=.25 and .25](O,A)
\tkzDrawSegments(O,P)
\tkzDrawPoints(A,P,O) \tkzDrawPoints[new](A',P')
\end{tikzpicture}
```



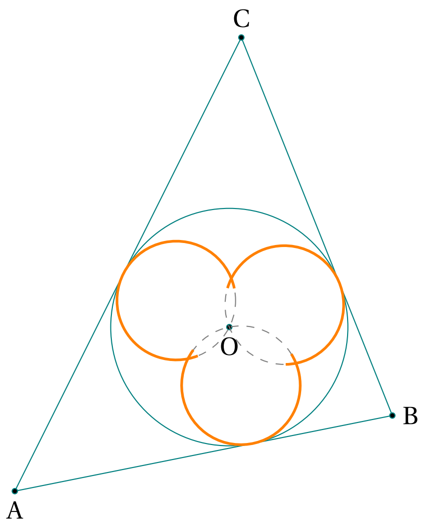
```
\begin{tikzpicture}[scale=.5]
\tkzDefPoints{0/0/O,3/0/I,2/1/P,2/-2/Q}
\tkzDrawCircle(O,I)
\tkzDefPointBy[projection= onto P--Q](O) \tkzGetPoint{A}
\tkzDefPointBy[inversion = center O through I](A)
\tkzGetPoint{A'}
\tkzDefPointBy[inversion = center O through I](P)
\tkzGetPoint{P'}
\tkzDrawCircle[new,diameter](O,A')
\tkzDrawLines[add=.25 and .75,red](P,Q)
\tkzDrawLines[add=.25 and .25](O,A')
\tkzDrawSegments(O,P')
\tkzDrawPoints(A,P,O) \tkzDrawPoints[new](A',P')
\end{tikzpicture}
```

12.1.10. Inversion of circle




```
\begin{tikzpicture}[scale=.5]
\tkzDefPoints{0/0/O,3/2/A,2/1/P}
\tkzDefTangent[from = O](A,P) \tkzGetPoints{T}{X}
\tkzDefPointsBy[homothety=center O ratio 1.25](A,P,T){}
\tkzInterCC(A,P)(A',P') \tkzGetPoints{C}{D}
\tkzCalcLength(A,P)
\tkzGetLength{rAP}
\tkzDefPointOnCircle[angle=190,center=A,radius=\rAP]
\tkzGetPoint{M}
\tkzDefPointBy[inversion = center O through C](M)
\tkzGetPoint{M'}
\tkzDrawCircles(A,P A',P')
\tkzDrawCircle(O,C)
\tkzDrawLines[add=0 and .5](O,T' O,A' O,M' O,P')
\tkzDrawPoints(A,A',P,P',O,T,T',M,M')
\tkzLabelPoints(O,T,T')
\tkzLabelPoints[above left](M,M')
\tkzLabelPoints[below](P,P')
\end{tikzpicture}
```

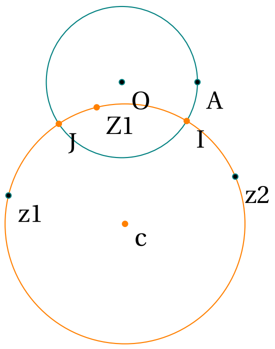
12.1.11. Inversion of Triangle with respect to the Incircle




```
\begin{tikzpicture}[scale=1]
\tkzDefPoints{0/0/A,5/1/B,3/6/C}
\tkzDefTriangleCenter[in](A,B,C) \tkzGetPoint{O}
\tkzDefPointBy[projection= onto A--C](O) \tkzGetPoint{b}
\tkzDefPointBy[projection= onto A--C](O) \tkzGetPoint{b}
\tkzDefPointBy[projection= onto B--C](O) \tkzGetPoint{a}
\tkzDefPointBy[projection= onto A--B](O) \tkzGetPoint{c}
\tkzDefPointsBy[inversion = center O through b](a,b,c)%
                                             {Ia,Ib,Ic}
\tkzDefMidPoint(O,Ia) \tkzGetPoint{Ja}
\tkzDefMidPoint(O,Ib) \tkzGetPoint{Jb}
\tkzDefMidPoint(O,Ic) \tkzGetPoint{Jc}
\tkzInterCC(Ja,O)(Jb,O) \tkzGetPoints{O}{x}
\tkzInterCC(Ja,O)(Jc,O) \tkzGetPoints{y}{O}
\tkzInterCC(Jb,O)(Jc,O) \tkzGetPoints{O}{z}
\tkzDrawPolygon(A,B,C)
\tkzDrawCircle(O,b)\tkzDrawPoints(A,B,C,O)
\tkzDrawCircles[dashed,gray](Ja,y Jb,x Jc,z)
\tkzDrawArc[line width=1pt,orange](Jb,x)(z)
\tkzDrawArc[line width=1pt,orange](Jc,z)(y)
\tkzDrawArc[line width=1pt,orange](Ja,y)(x)
\tkzLabelPoint[below](A){$A$}\tkzLabelPoint[above](C){$C$}
\tkzLabelPoint[right](B){$B$}\tkzLabelPoint[below](O){$O$}
\end{tikzpicture}
```

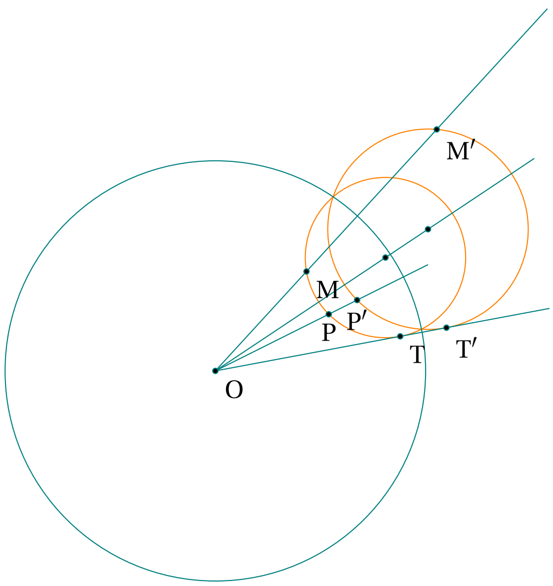
12.1.12. Inversion: orthogonal circle with inversion circle

The inversion circle itself, circles orthogonal to it, and lines through the inversion center are invariant under inversion. If the circle meets the reference circle, these invariant points of intersection are also on the inverse circle. See I and J in the next figure.




```
\begin{tikzpicture}[scale=1]
\tkzDefPoint(0,0){O}\tkzDefPoint(1,0){A}
\tkzDefPoint(-1.5,-1.5){z1}
\tkzDefPoint(1.5,-1.25){z2}
\tkzDefCircleBy[orthogonal through=z1 and z2](O,A)
\tkzGetPoint{c}
\tkzDrawCircle[new](c,z1)
\tkzDefPointBy[inversion =  center O through A](z1)
\tkzGetPoint{Z1}
\tkzInterCC(O,A)(c,z1) \tkzGetPoints{I}{J}
\tkzDefPointBy[inversion =  center O through A](I)
\tkzGetPoint{I'}
\tkzDrawCircle(O,A)
\tkzDrawPoints(O,A,z1,z2)
\tkzDrawPoints[new](c,Z1,I,J)
\tkzLabelPoints(O,A,z1,z2,c,Z1,I,J)
\end{tikzpicture}
```

12.1.13. Inversion and homothety

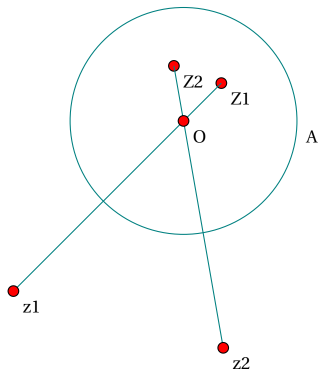



```
\begin{tikzpicture}[scale=.75]
\tkzDefPoints{0/0/O,3/2/A,2/1/P}
\tkzDefTangent[from = O](A,P) \tkzGetPoints{T}{X}
\tkzDefPointsBy[homothety = center O%
                ratio 1.25](A,P,T){}
\tkzInterCC(A,P)(A',P') \tkzGetPoints{C}{D}
\tkzCalcLength(A,P)
\tkzGetLength{rAP}
\tkzDefPointOnCircle[angle=190,center=A,radius=\rAP]
\tkzGetPoint{M}
\tkzDefPointBy[inversion = center O through C](M)
\tkzGetPoint{M'}
\tkzDrawCircles[new](A,P A',P')
\tkzDrawCircle(O,C)
\tkzDrawLines[add=0 and .5](O,T' O,A' O,M' O,P')
\tkzDrawPoints(A,A',P,P',O,T,T',M,M')
\tkzLabelPoints(O,T,T',M,M')
\tkzLabelPoints[below](P,P')
\end{tikzpicture}
```

For a more complex example see **Pappus** 46.24

12.1.14. inversion negative

It’s an inversion followed by a symmetry of center O




```
\begin{tikzpicture}[scale=1.5]
  \tkzDefPoints{1/0/A,0/0/O}
  \tkzDefPoint(-1.5,-1.5){z1}
  \tkzDefPoint(0.35,-2){z2}
  \tkzDefPointBy[inversion negative = center O through A](z1)
  \tkzGetPoint{Z1}
  \tkzDefPointBy[inversion negative = center O through A](z2)
  \tkzGetPoint{Z2}
  \tkzDrawCircle(O,A)
  \tkzDrawPoints[color=black, fill=red,size=4](Z1,Z2)
  \tkzDrawSegments(z1,Z1 z2,Z2)
  \tkzDrawPoints[color=black, fill=red,size=4](O,z1,z2)
  \tkzLabelPoints[font=\scriptsize](O,A,z1,z2,Z1,Z2)
\end{tikzpicture}
```

12.2. Transformation of multiple points; \tkzDefPointsBy

Variant of the previous macro for defining multiple images. You must give the names of the images as arguments, or indicate that the names of the images are formed from the names of the antecedents, leaving the argument empty.

```
\tkzDefPointsBy[translation= from A to A'](B,C){}
```

The images are B' and C'.

```
\tkzDefPointsBy[translation= from A to A'](B,C){D,E}
```

The images are D and E.

```
\tkzDefPointsBy[translation= from A to A'](B)
```

The image is B'.

`\tkzDefPointsBy[⟨local options⟩](⟨list of points⟩){⟨list of points⟩}`

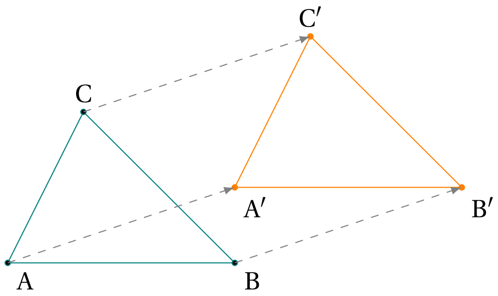
arguments	examples	
`(⟨list of points⟩){⟨list of pts⟩}`	`(A,B){E,F}`	`E,F images of A, B`

If the list of images is empty then the name of the image is the name of the antecedent to which " ' " is added.

options	examples
`translation = from #1 to #2`	`[translation=from A to B](E){}`
`homothety = center #1 ratio #2`	`[homothety=center A ratio .5](E){F}`
`reflection = over #1--#2`	`[reflection=over A--B](E){F}`
`symmetry = center #1`	`[symmetry=center A](E){F}`
`projection = onto #1--#2`	`[projection=onto A--B](E){F}`
`rotation = center #1 angle #2`	`[rotation=center angle 30](E){F}`
`rotation in rad = center #1 angle #2`	`for instance angle pi/3`

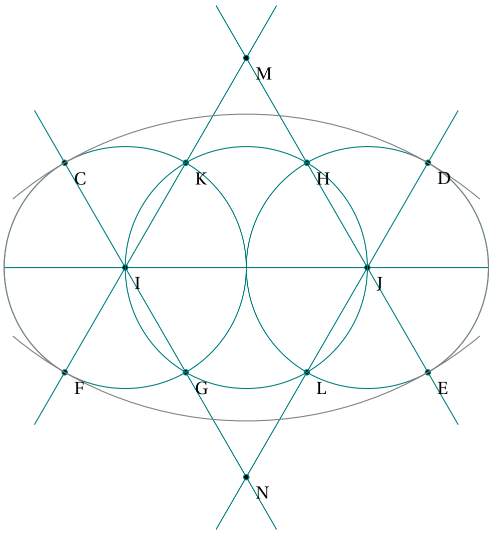
The points are only defined and not drawn.

12.2.1. Example of translation




```
\begin{tikzpicture}[>=latex]
 \tkzDefPoints{0/0/A,3/0/B,3/1/A',1/2/C}
 \tkzDefPointsBy[translation= from A to A'](B,C){}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawPolygon[new](A',B',C')
 \tkzDrawPoints(A,B,C)
 \tkzDrawPoints[new](A',B',C')
 \tkzLabelPoints(A,B,A',B')
 \tkzLabelPoints[above](C,C')
 \tkzDrawSegments[color = gray,->,
             style=dashed](A,A' B,B' C,C')
\end{tikzpicture}
```

12.2.2. Example of symmetry




```
\begin{tikzpicture}[scale=.4]
  \tkzDefPoint(-4,0){I}
  \tkzDefPoint(4,0){J}
  \tkzDefPoint(0,0){O}
  \tkzInterCC(J,O)(O,J) \tkzGetPoints{L}{H}
  \tkzInterCC(I,O)(O,I) \tkzGetPoints{K}{G}
  \tkzDrawLines[add=1.5 and 1.5](I,K I,G J,H J,L)
  \tkzDrawLines[add=.5 and .5](I,J)
  \tkzInterLL(I,K)(J,H) \tkzGetPoint{M}
  \tkzInterLL(I,G)(J,L) \tkzGetPoint{N}
  \tkzDefPointsBy[symmetry=center J](L,H){D,E}
  \tkzDefPointsBy[symmetry=center I](G,K){C,F}
  \tkzDrawPoints(H,L,K,G,I,J,D,E,C,F,M,N)
  \tkzDrawCircle[R](O,4)
  \tkzDrawCircle[R](I,4)
  \tkzDrawCircle[R](J,4)
  \tkzDrawArc(N,D)(C)
  \tkzDrawArc(M,F)(E)
  \tkzDrawArc(J,E)(D)
  \tkzDrawArc(I,C)(F)
  \tkzLabelPoints[font=\scriptsize](H,L,K,G,I,J,%
                                    D,E,C,F,M,N)
\end{tikzpicture}
```

13. Defining points using a vector

13.1. \tkzDefPointWith

There are several possibilities to create points that meet certain vector conditions. This can be done with `\tkzDefPointWith`. The general principle is as follows, two points are passed as arguments, i.e. a vector. The different options allow to obtain a new point forming with the first point (with some exceptions) a collinear vector or a vector orthogonal to the first vector. Then the length is either proportional to that of the first one, or proportional to the unit. Since this point is only used temporarily, it does not have to be named immediately. The result is in `tkzPointResult`. The macro `\tkzGetPoint` allows you to retrieve the point and name it differently. There are options to define the distance between the given point and the obtained point. In the general case this distance is the distance between the 2 points given as arguments if the option is of the "normed" type then the distance between the given point and the obtained point is 1 cm. Then the K option allows to obtain multiples.

`\tkzDefPointWith(⟨pt1,pt2⟩)`

It is in fact the definition of a point meeting vectorial conditions.

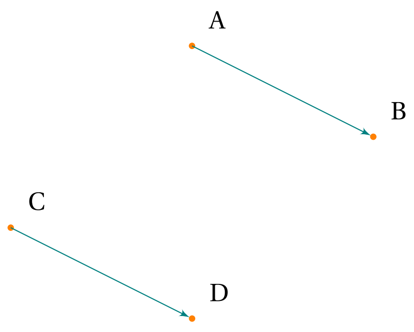
arguments	definition	explication
(pt1,pt2)	point couple	the result is a point in **tkzPointResult**

In what follows, it is assumed that the point is recovered by `\tkzGetPoint{C}`

options	example	explication
orthogonal	[orthogonal](A,B)	$AC = AB$ and $\overrightarrow{AC} \perp \overrightarrow{AB}$
orthogonal normed	[orthogonal normed](A,B)	$AC = 1$ and $\overrightarrow{AC} \perp \overrightarrow{AB}$
linear	[linear](A,B)	$\overrightarrow{AC} = K \times \overrightarrow{AB}$
linear normed	[linear normed](A,B)	$AC = K$ and $\overrightarrow{AC} = k \times \overrightarrow{AB}$
colinear= at #1	[colinear= at C](A,B)	$\overrightarrow{CD} = \overrightarrow{AB}$
colinear normed= at #1	[colinear normed= at C](A,B)	$\overrightarrow{CD} = \overrightarrow{AB}$
K	[linear](A,B),K=2	$\overrightarrow{AC} = 2 \times \overrightarrow{AB}$

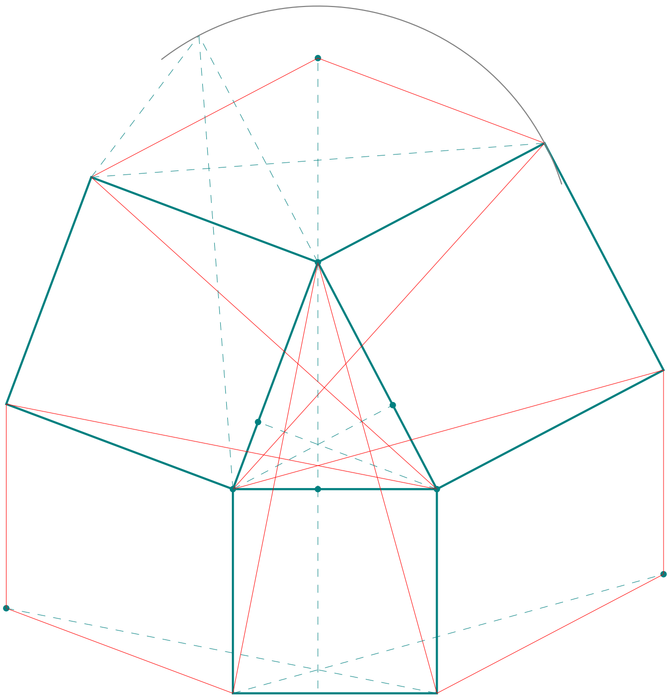
13.1.1. Option colinear at, simple example

($\overrightarrow{AB} = \overrightarrow{CD}$)




```
\begin{tikzpicture}[scale=1.2,
   vect/.style={->,shorten >=1pt,>=latex'}]
  \tkzDefPoint(2,3){A}   \tkzDefPoint(4,2){B}
  \tkzDefPoint(0,1){C}
  \tkzDefPointWith[colinear=at C](A,B)
  \tkzGetPoint{D}
  \tkzDrawPoints[new](A,B,C,D)
  \tkzLabelPoints[above right=3pt](A,B,C,D)
  \tkzDrawSegments[vect](A,B C,D)
\end{tikzpicture}
```

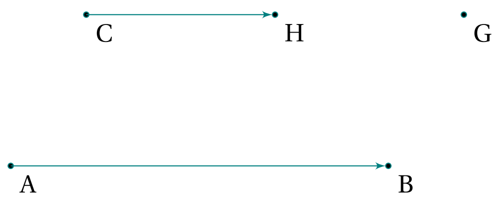
13.1.2. Option colinear at, complex example



```
\begin{tikzpicture}[scale=.75]
\tkzDefPoints{0/0/B,3.6/0/C,1.5/4/A}
\tkzDefSpcTriangle[ortho](A,B,C){Ha,Hb,Hc}
\tkzDefTriangleCenter[ortho](A,B,C) \tkzGetPoint{H}
\tkzDefSquare(A,C) \tkzGetPoints{R}{S}
\tkzDefSquare(B,A) \tkzGetPoints{M}{N}
\tkzDefSquare(C,B) \tkzGetPoints{P}{Q}
\tkzDefPointWith[colinear= at M](A,S) \tkzGetPoint{A'}
\tkzDefPointWith[colinear= at P](B,N) \tkzGetPoint{B'}
\tkzDefPointWith[colinear= at Q](C,R) \tkzGetPoint{C'}
\tkzDefPointBy[projection=onto P--Q](Ha) \tkzGetPoint{Pa}
\tkzDrawPolygon[teal,thick](A,C,R,S)\tkzDrawPolygon[teal,thick](A,B,N,M)
\tkzDrawPolygon[teal,thick](C,B,P,Q)
\tkzDrawPoints[teal,size=2](A,B,C,Ha,Hb,Hc,A',B',C')
\tkzDrawSegments[ultra thin,red](M,A' A',S P,B' B',N Q,C' C',R B,S C,M C,N B,R A,P A,Q)
\tkzDrawSegments[ultra thin,teal, dashed](A,Ha B,Hb C,Hc)
\tkzDefPointBy[rotation=center A angle 90](S) \tkzGetPoint{S'}
\tkzDrawSegments[ultra thin,teal,dashed](B,S' A,S' A,A' M,S' B',Q P,C' M,S Ha,Pa)
\tkzDrawArc(A,S)(S')
\end{tikzpicture}
```

13.1.3. Option colinear at

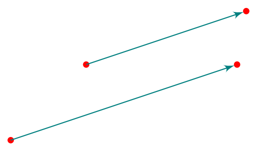
How to use K




```
\begin{tikzpicture}[vect/.style={->,
               shorten >=1pt,>=latex'}]
  \tkzDefPoints{0/0/A,5/0/B,1/2/C}
  \tkzDefPointWith[colinear=at C](A,B)
  \tkzGetPoint{G}
  \tkzDefPointWith[colinear=at C, K=0.5](A,B)
  \tkzGetPoint{H}
  \tkzLabelPoints(A,B,C,G,H)
  \tkzDrawPoints(A,B,C,G,H)
  \tkzDrawSegments[vect](A,B C,H)
\end{tikzpicture}
```

13.1.4. Option `colinear at`

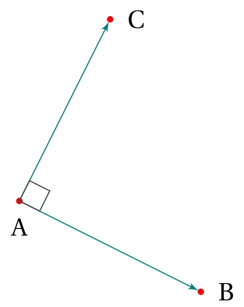
With $K = \frac{\sqrt{2}}{2}$



```
\begin{tikzpicture}[vect/.style={->,
            shorten >=1pt,>=latex'}]
 \tkzDefPoints{1/1/A,4/2/B,2/2/C}
 \tkzDefPointWith[colinear=at C,K=sqrt(2)/2](A,B)
 \tkzGetPoint{D}
 \tkzDrawPoints[color=red](A,B,C,D)
 \tkzDrawSegments[vect](A,B C,D)
\end{tikzpicture}
```

13.1.5. Option `orthogonal`

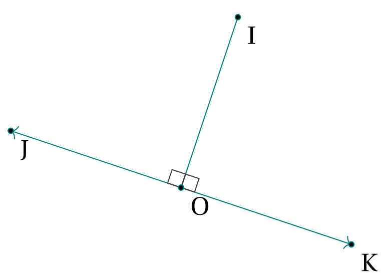
AB=AC since $K = 1$.




```
\begin{tikzpicture}[scale=1.2,
  vect/.style={->,shorten >=1pt,>=latex'}]
  \tkzDefPoints{2/3/A,4/2/B}
   \tkzDefPointWith[orthogonal,K=1](A,B)
     \tkzGetPoint{C}
   \tkzDrawPoints[color=red](A,B,C)
   \tkzLabelPoints[right=3pt](B,C)
   \tkzLabelPoints[below=3pt](A)
   \tkzDrawSegments[vect](A,B A,C)
   \tkzMarkRightAngle(B,A,C)
\end{tikzpicture}
```

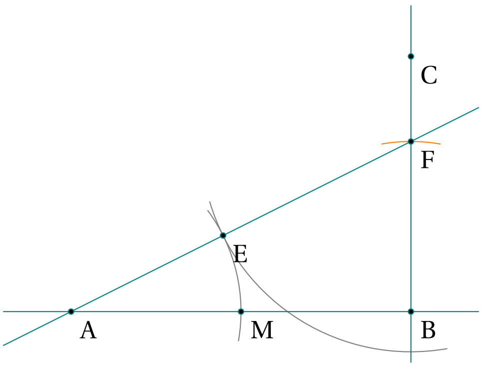
13.1.6. Option `orthogonal`

With $K = -1$ OK=OI since $|K| = 1$ then OI=OJ=OK.



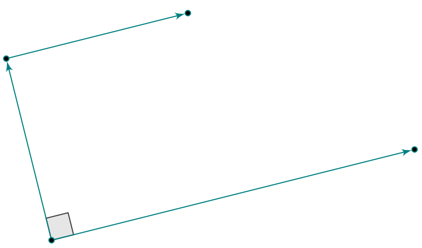

```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoints{1/2/O,2/5/I}
  \tkzDefPointWith[orthogonal](O,I)
  \tkzGetPoint{J}
  \tkzDefPointWith[orthogonal,K=-1](O,I)
  \tkzGetPoint{K}
  \tkzDrawSegment(O,I)
  \tkzDrawSegments[->](O,J O,K)
  \tkzMarkRightAngles(I,O,J I,O,K)
  \tkzDrawPoints(O,I,J,K)
  \tkzLabelPoints(O,I,J,K)
\end{tikzpicture}
```

13.1.7. Option orthogonal more complicated example




```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoints{0/0/A,6/0/B}
  \tkzDefMidPoint(A,B)
    \tkzGetPoint{I}
  \tkzDefPointWith[orthogonal,K=-.75](B,A)
  \tkzGetPoint{C}
  \tkzInterLC(B,C)(B,I)
     \tkzGetPoints{D}{F}
  \tkzDuplicateSegment(B,F)(A,F)
  \tkzGetPoint{E}
  \tkzDrawArc[delta=10](F,E)(B)
  \tkzInterLC(A,B)(A,E)
    \tkzGetPoints{N}{M}
  \tkzDrawArc[delta=10](A,M)(E)
  \tkzDrawLines(A,B B,C A,F)
  \tkzCompass(B,F)
  \tkzDrawPoints(A,B,C,F,M,E)
  \tkzLabelPoints(A,B,C,F,M,E)
\end{tikzpicture}
```

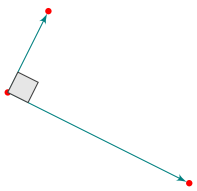
13.1.8. Options colinear and orthogonal



```
\begin{tikzpicture}[scale=1.2,
  vect/.style={->,shorten >=1pt,>=latex'}]
  \tkzDefPoints{2/1/A,6/2/B}
  \tkzDefPointWith[orthogonal,K=.5](A,B)
  \tkzGetPoint{C}
  \tkzDefPointWith[colinear=at C,K=.5](A,B)
  \tkzGetPoint{D}
  \tkzMarkRightAngle[fill=gray!20](B,A,C)
  \tkzDrawSegments[vect](A,B A,C C,D)
  \tkzDrawPoints(A,...,D)
\end{tikzpicture}
```

13.1.9. Option orthogonal normed

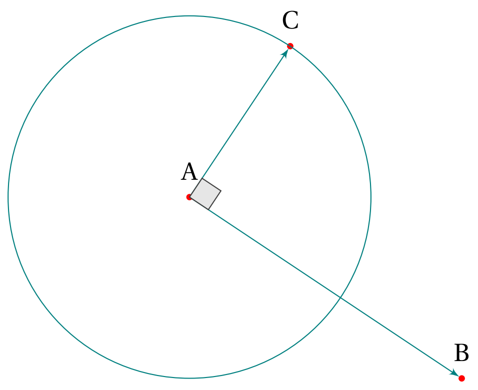
K = 1 AC = 1.



```
\begin{tikzpicture}[scale=1.2,
  vect/.style={->,shorten >=1pt,>=latex'}]
  \tkzDefPoints{2/3/A,4/2/B}
  \tkzDefPointWith[orthogonal normed](A,B)
  \tkzGetPoint{C}
  \tkzDrawPoints[color=red](A,B,C)
  \tkzDrawSegments[vect](A,B A,C)
  \tkzMarkRightAngle[fill=gray!20](B,A,C)
\end{tikzpicture}
```

13.1.10. Option orthogonal normed and K=2

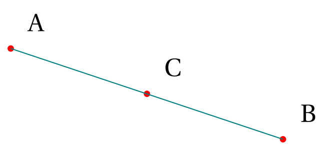
K = 2 therefore AC = 2.




```
\begin{tikzpicture}[scale=1.2,
   vect/.style={->,shorten >=1pt,>=latex'}]
  \tkzDefPoints{2/3/A,5/1/B}
  \tkzDefPointWith[orthogonal normed,K=2](A,B)
  \tkzGetPoint{C}
  \tkzDrawPoints[color=red](A,B,C)
  \tkzDrawCircle[R](A,2)
  \tkzDrawSegments[vect](A,B A,C)
  \tkzMarkRightAngle[fill=gray!20](B,A,C)
  \tkzLabelPoints[above=3pt](A,B,C)
\end{tikzpicture}
```

13.1.11. Option linear

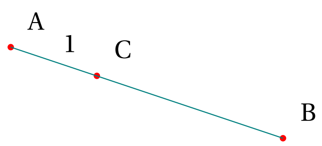
Here K = 0.5.
This amounts to applying a homothety or a multiplication of a vector by a real. Here is the middle of [AB].




```
\begin{tikzpicture}[scale=1.2]
  \tkzDefPoints{1/3/A,4/2/B}
  \tkzDefPointWith[linear,K=0.5](A,B)
  \tkzGetPoint{C}
  \tkzDrawPoints[color=red](A,B,C)
  \tkzDrawSegment(A,B)
  \tkzLabelPoints[above right=3pt](A,B,C)
\end{tikzpicture}
```

13.1.12. Option linear normed

In the following example AC = 1 and C belongs to (AB).




```
\begin{tikzpicture}[scale=1.2]
 \tkzDefPoints{1/3/A,4/2/B}
 \tkzDefPointWith[linear normed](A,B)
 \tkzGetPoint{C}
 \tkzDrawPoints[color=red](A,B,C)
 \tkzDrawSegment(A,B)
 \tkzLabelSegment(A,C){$1$}
 \tkzLabelPoints[above right=3pt](A,B,C)
\end{tikzpicture}
```

13.2. \tkzGetVectxy

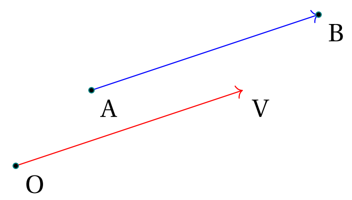
Retrieving the coordinates of a vector.

\tkzGetVectxy(⟨A,B⟩){⟨text⟩}

Allows to obtain the coordinates of a vector.

arguments	example	explication
(point){name of macro}	\tkzGetVectxy(A,B){V}	\Vx,\Vy: coordinates of $\overrightarrow{AB}$

13.2.1. Coordinate transfer with \tkzGetVectxy




```
\begin{tikzpicture}
 \tkzDefPoints{0/0/O,1/1/A,4/2/B}
 \tkzGetVectxy(A,B){v}
 \tkzDefPoint(\vx,\vy){V}
 \tkzDrawSegment[->,color=red](O,V)
 \tkzDrawSegment[->,color=blue](A,B)
 \tkzDrawPoints(A,B,O)
 \tkzLabelPoints(A,B,O,V)
\end{tikzpicture}
```

14. The straight lines

It is of course essential to draw straight lines, but before this can be done, it is necessary to be able to define certain particular lines such as mediators, bisectors, parallels or even perpendiculars. The principle is to determine two points on the straight line.

14.1. Definition of straight lines

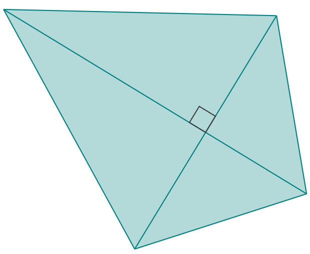
`\tkzDefLine[⟨local options⟩](⟨pt1,pt2⟩) or (⟨pt1,pt2,pt3⟩)`

The argument is a list of two or three points. Depending on the case, the macro defines one or two points necessary to obtain the line sought. Either the macro `\tkzGetPoint` or the macro `\tkzGetPoints` must be used. I used the term "mediator" to designate the perpendicular bisector line at the middle of a line segment.

arguments	example	explication
(⟨pt1,pt2⟩)	(⟨A,B⟩)	[mediator](A,B)
(⟨pt1,pt2,pt3⟩)	(⟨A,B,C⟩)	[bisector](B,A,C)

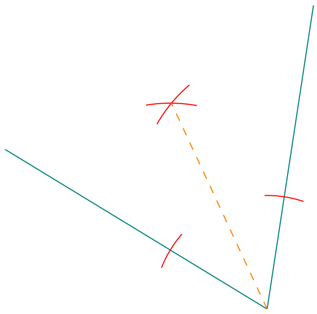
options	default	definition
mediator		perpendicular bisector of a line segment
perpendicular=through...	mediator	perpendicular to a straight line passing through a point
orthogonal=through...	mediator	see above
parallel=through...	mediator	parallel to a straight line passing through a point
bisector	mediator	bisector of an angle defined by three points
bisector out	mediator	Exterior Angle Bisector
K	1	coefficient for the perpendicular line
normed	false	normalizes the created segment

14.1.1. Example with mediator



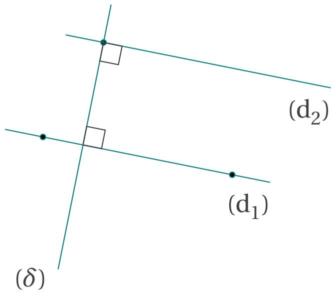
```
\begin{tikzpicture}[rotate=25]
 \tkzDefPoints{-2/0/A,1/2/B}
 \tkzDefLine[mediator](A,B)           \tkzGetPoints{C}{D}
 \tkzDefPointWith[linear,K=.75](C,D)  \tkzGetPoint{D}
 \tkzDefMidPoint(A,B)                 \tkzGetPoint{I}
 \tkzFillPolygon[color=teal!30](A,C,B,D)
 \tkzDrawSegments(A,B C,D)
 \tkzMarkRightAngle(B,I,C)
 \tkzDrawSegments(D,B D,A)
 \tkzDrawSegments(C,B C,A)
\end{tikzpicture}
```

14.1.2. Example with bisector and normed



```
\begin{tikzpicture}[rotate=25,scale=.75]
 \tkzDefPoints{0/0/C, 2/-3/A, 4/0/B}
 \tkzDefLine[bisector,normed](B,A,C) \tkzGetPoint{a}
 \tkzDrawLines[add= 0 and .5](A,B A,C)
 \tkzShowLine[bisector,gap=4,size=2,color=red](B,A,C)
 \tkzDrawLines[new,dashed,add= 0 and 3](A,a)
\end{tikzpicture}
```

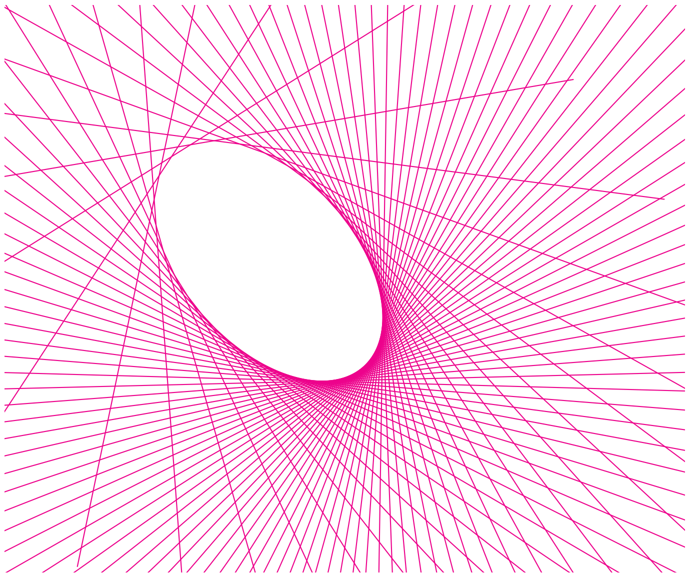
14.1.3. Example with orthogonal and parallel




```
\begin{tikzpicture}
   \tkzDefPoints{-1.5/-0.25/A,1/-0.75/B,-0.7/1/C}
   \tkzDrawLine(A,B)
   \tkzLabelLine[pos=1.25,below left](A,B){$(d_1)$}
   \tkzDrawPoints(A,B,C)
   \tkzDefLine[orthogonal=through C](B,A) \tkzGetPoint{c}
   \tkzDrawLine(C,c)
   \tkzLabelLine[pos=1.25,left](C,c){$(\delta)$}
   \tkzInterLL(A,B)(C,c) \tkzGetPoint{I}
   \tkzMarkRightAngle(C,I,B)
   \tkzDefLine[parallel=through C](A,B) \tkzGetPoint{c'}
   \tkzDrawLine(C,c')
   \tkzLabelLine[pos=1.25,below left](C,c'){$(d_2)$}
   \tkzMarkRightAngle(I,C,c')
\end{tikzpicture}
```

14.1.4. An envelope

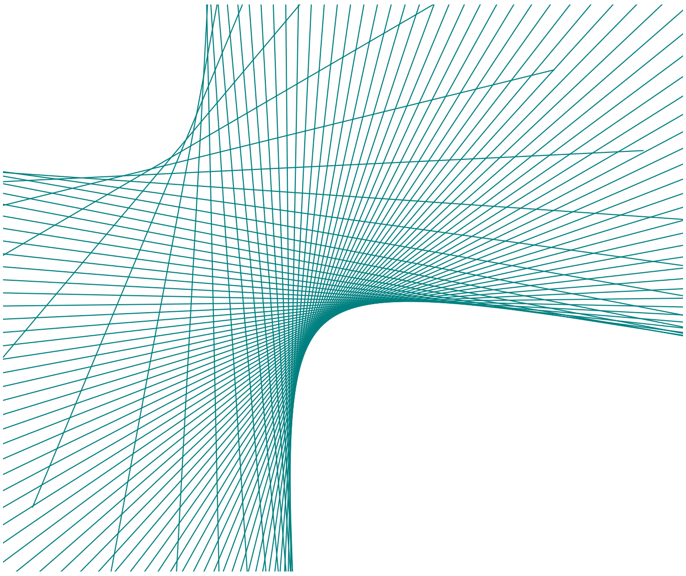
Based on a figure from O. Reboux with pst-eucl by D Rodriguez.



```
\begin{tikzpicture}[scale=.75]
 \tkzInit[xmin=-6,ymin=-4,xmax=6,ymax=6] % necessary
 \tkzClip
 \tkzDefPoint(0,0){O}
 \tkzDefPoint(132:4){A}
 \tkzDefPoint(5,0){B}
 \foreach \ang in {5,10,...,360}{%
  \tkzDefPoint(\ang:5){M}
  \tkzDefLine[mediator](A,M)
  \tkzDrawLine[color=magenta,add= 3 and 3](tkzFirstPointResult,tkzSecondPointResult)}
\end{tikzpicture}
```

14.1.5. A parabola

Based on a figure from O. Reboux with pst-eucl by D Rodriguez. It is not necessary to name the two points that define the mediator.



```
\begin{tikzpicture}[scale=.75]
 \tkzInit[xmin=-6,ymin=-4,xmax=6,ymax=6]
 \tkzClip
 \tkzDefPoint(0,0){O}
 \tkzDefPoint(132:5){A}
 \tkzDefPoint(4,0){B}
 \foreach \ang in {5,10,...,360}{%
  \tkzDefPoint(\ang:4){M}
  \tkzDefLine[mediator](A,M)
  \tkzDrawLine[color=teal,add= 3 and 3](tkzFirstPointResult,tkzSecondPointResult)}
\end{tikzpicture}
```

14.2. Specific lines: Tangent to a circle

Two constructions are proposed. The first one is the construction of a tangent to a circle at a given point of this circle and the second one is the construction of a tangent to a circle passing through a given point outside a disc.

`\tkzDefTangent[⟨local options⟩](⟨pt1,pt2⟩) or (⟨pt1,dim⟩)`

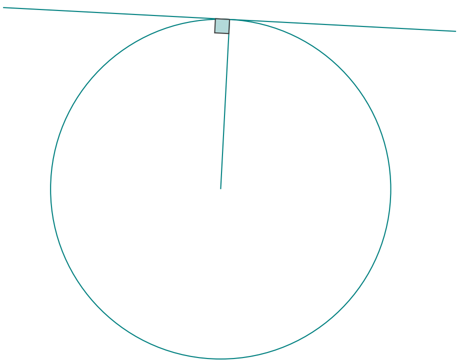
The parameter in brackets is the center of the circle or the center of the circle and a point on the circle or the center and the radius. This macro replaces the old one: **\tkzTangent**.

arguments	example	explication
(⟨pt1,pt2 or (⟨pt1,dim⟩)⟩)	(⟨A,B⟩) or (⟨A,2cm⟩)	[AB] is radius A is the center

options	default	definition
at=pt	at	tangent to a point on the circle
from=pt	at	tangent to a circle passing through a point
from with R=pt	at	idem, but the circle is defined by center = radius

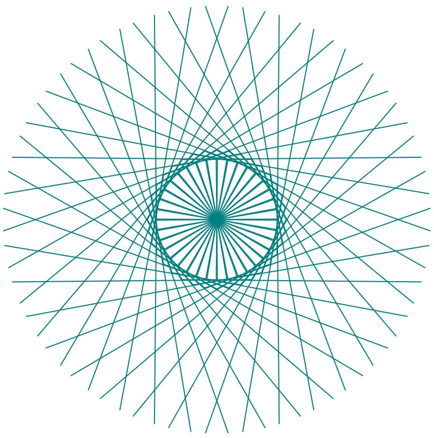
The tangent is not drawn. A second point of the tangent is given by **tkzPointResult**.

14.2.1. Example of a tangent passing through a point on the circle



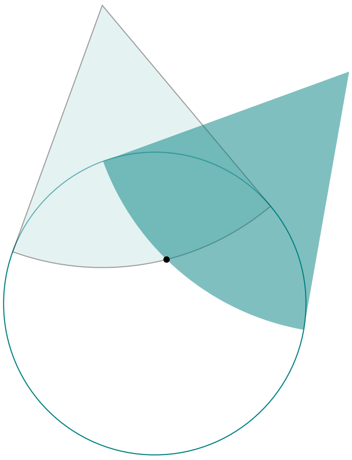
```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(6,6){E}
  \tkzDefRandPointOn[circle=center O radius 3]
  \tkzGetPoint{A}
  \tkzDrawSegment(O,A)
  \tkzDrawCircle(O,A)
  \tkzDefTangent[at=A](O)
  \tkzGetPoint{h}
  \tkzDrawLine[add = 4 and 3](A,h)
  \tkzMarkRightAngle[fill=teal!30](O,A,h)
\end{tikzpicture}
```

14.2.2. Example of tangents passing through an external point



```
\begin{tikzpicture}[scale=.8]
  \tkzDefPoint(3,3){c}
  \tkzDefPoint(6,3){a0}
  \pgfmathsetmacro\R{1}
  \tkzDrawCircle[R](c,\R)
  \foreach \an in {0,10,...,350}{
     \tkzDefPointBy[rotation=center c angle \an](a0)
     \tkzGetPoint{a}
     \tkzDefTangent[from with R = a](c,\R)
     \tkzGetPoints{e}{f}
     \tkzDrawLines[color=teal](a,f a,e)
     \tkzDrawSegments(c,e c,f)
  }%
\end{tikzpicture}
```

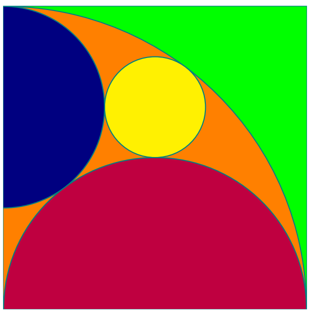
14.2.3. Example of Andrew Mertz



```
\begin{tikzpicture}[scale=.5]
\tkzDefPoint(100:8){A}\tkzDefPoint(50:8){B}
\tkzDefPoint(0,0){C} \tkzDefPoint(0,4){R}
\tkzDrawCircle(C,R)
\tkzDefTangent[from = A](C,R)  \tkzGetPoints{D}{E}
\tkzDefTangent[from = B](C,R)  \tkzGetPoints{F}{G}
\tkzDrawSector[fill=teal!20,opacity=0.5](A,D)(E)
\tkzFillSector[color=teal,opacity=0.5](B,F)(G)
\tkzInterCC(A,D)(B,F) \tkzGetSecondPoint{I}
\tkzDrawPoint[color=black](I)
\end{tikzpicture}
```

http://www.texample.net/tikz/examples/

14.2.4. Drawing a tangent option from



```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoint(0,0){B}
 \tkzDefPoint(0,8){A}
 \tkzDefSquare(A,B)
 \tkzGetPoints{C}{D}
 \tkzDrawSquare(A,B)
 \tkzClipPolygon(A,B,C,D)
 \tkzDefPoint(4,8){F}
 \tkzDefPoint(4,0){E}
 \tkzDefPoint(4,4){Q}
 \tkzFillPolygon[color = green](A,B,C,D)
 \tkzDrawCircle[fill = orange](B,A)
 \tkzDrawCircle[fill = purple](E,B)
 \tkzDefTangent[from=B](F,A)
 \tkzInterLL(F,tkzFirstPointResult)(C,D)
 \tkzInterLL(A,tkzPointResult)(F,E)
 \tkzDrawCircle[fill = yellow](tkzPointResult,Q)
 \tkzDefPointBy[projection= onto B--A](tkzPointResult)
 \tkzDrawCircle[fill = blue!50!black](tkzPointResult,A)
\end{tikzpicture}
```

15. Triangles

15.1. Definition of triangles \tkzDefTriangle

The following macros will allow you to define or construct a triangle from **at least** two points.
At the moment, it is possible to define the following triangles:

- **two angles** determines a triangle with two angles;
- **equilateral** determines an equilateral triangle;
- **isosceles right** determines an isoxsceles right triangle;
- **half** determines a right-angled triangle such that the ratio of the measurements of the two adjacent sides to the right angle is equal to 2;
- **pythagore** determines a right-angled triangle whose side measurements are proportional to 3, 4 and 5;
- **school** determines a right-angled triangle whose angles are 30, 60 and 90 degrees;
- **golden** determines a right-angled triangle such that the ratio of the measurements on the two adjacent sides to the right angle is equal to $\Phi = 1.618034$, I chose "golden triangle" as the denomination because it comes from the golden rectangle and I kept the denomination "gold triangle" or "Euclid's triangle" for the isosceles triangle whose angles at the base are 72 degrees;
- **euclid** or **gold** for the gold triangle; in the previous version the option was "euclide" with an "e".
- **cheops** determines a third point such that the triangle is isosceles with side measurements proportional to 2, Φ and Φ.

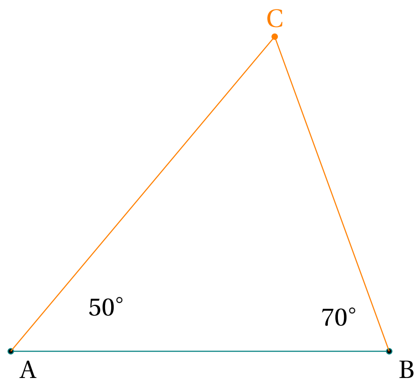
`\tkzDefTriangle[⟨local options⟩](⟨A,B⟩)`

The points are ordered because the triangle is constructed following the direct direction of the trigonometric circle. This macro is either used in partnership with `\tkzGetPoint` or by using `tkzPointResult` if it is not necessary to keep the name.

options	default	definition
two angles= #1 and #2	no defaut	triangle knowing two angles
equilateral	equilateral	equilateral triangle
isosceles right	equilateral	isosceles right triangle
pythagore	equilateral	proportional to the pythagorean triangle 3-4-5
school	equilateral	angles of 30, 60 and 90 degrees
gold	equilateral	angles of 72, 72 and 36 degrees, A is the apex
euclid	equilateral	same as above but [AB] is the base
golden	equilateral	B rectangle and $AB/AC = \Phi$
cheops	equilateral	AC=BC, AC and BC are proportional to 2 and Φ.

`\tkzGetPoint` allows you to store the point otherwise `tkzPointResult` allows for immediate use.

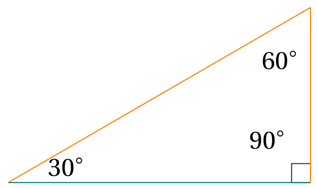
15.1.1. Option two angles




```
\begin{tikzpicture}
\tkzDefPoint(0,0){A}
\tkzDefPoint(5,0){B}
\tkzDefTriangle[two angles = 50 and 70](A,B)
\tkzGetPoint{C}
\tkzDrawSegment(A,B)
\tkzDrawPoints(A,B)
\tkzLabelPoints(A,B)
\tkzDrawSegments[new](A,C B,C)
\tkzDrawPoints[new](C)
\tkzLabelPoints[above,new](C)
\tkzLabelAngle[pos=1.4](B,A,C){$50^\circ$}
\tkzLabelAngle[pos=0.8](C,B,A){$70^\circ$}
\end{tikzpicture}
```

15.1.2. Option school

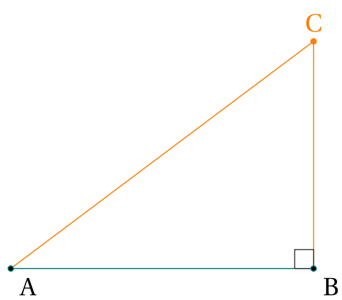
The angles are 30, 60 and 90 degrees.




```
\begin{tikzpicture}
  \tkzDefPoints{0/0/A,4/0/B}
  \tkzDefTriangle[school](A,B)
  \tkzGetPoint{C}
  \tkzMarkRightAngles(C,B,A)
  \tkzLabelAngle[pos=0.8](B,A,C){$30^\circ$}
  \tkzLabelAngle[pos=0.8](C,B,A){$90^\circ$}
  \tkzLabelAngle[pos=0.8](A,C,B){$60^\circ$}
  \tkzDrawSegments(A,B)
  \tkzDrawSegments[new](A,C B,C)
\end{tikzpicture}
```

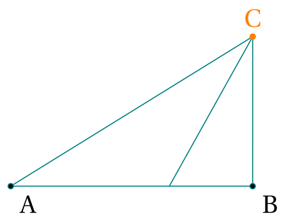
15.1.3. Option pythagore

This triangle has sides whose lengths are proportional to 3, 4 and 5.



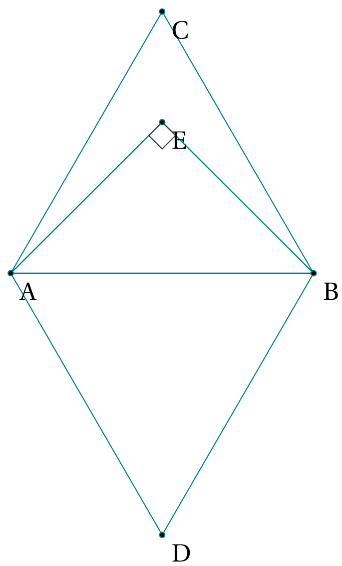

```
\begin{tikzpicture}
  \tkzDefPoints{0/0/A,4/0/B}
  \tkzDefTriangle[pythagore](A,B)
  \tkzGetPoint{C}
  \tkzDrawSegments(A,B)
  \tkzDrawSegments[new](A,C B,C)
  \tkzMarkRightAngles(A,B,C)
  \tkzLabelPoint[above,new](C){$C$}
  \tkzDrawPoints[new](C)
  \tkzDrawPoints(A,B)
  \tkzLabelPoints(A,B)
\end{tikzpicture}
```

15.1.4. Option golden



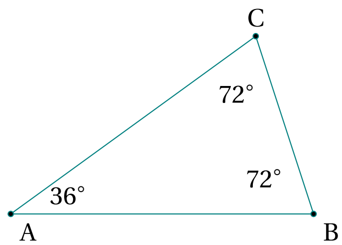

```
\begin{tikzpicture}[scale=.8]
\tkzDefPoint(0,0){A} \tkzDefPoint(4,0){B}
\tkzDefTriangle[golden](A,B)\tkzGetPoint{C}
\tkzDefSpcTriangle[in,name=M](A,B,C){a,b,c}
\tkzDrawPolygon(A,B,C)
\tkzDrawPoints(A,B)
\tkzDrawSegment(C,Mc)
\tkzDrawPoints[new](C)
\tkzLabelPoints(A,B)
\tkzLabelPoints[above,new](C)
\end{tikzpicture}
```

15.1.5. Option equilateral and isosceles right




```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(4,0){B}
  \tkzDefTriangle[equilateral](A,B)
  \tkzGetPoint{C}
  \tkzDefTriangle[isosceles right](A,B)
  \tkzGetPoint{E}
  \tkzDrawPolygons(A,B,C A,B,E)
  \tkzDefTriangle[equilateral](B,A)
  \tkzGetPoint{D}
  \tkzDrawPolygon(B,A,D)
    \tkzMarkRightAngles(B,E,A)
  \tkzDrawPoints(A,B,C,D,E)
  \tkzLabelPoints(A,B,C,D,E)
\end{tikzpicture}
```

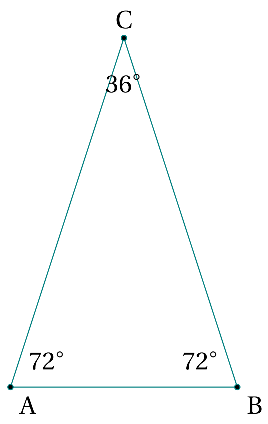
15.1.6. Option gold




```
\begin{tikzpicture}
  \tkzDefPoints{0/0/A,4/0/B}
  \tkzDefTriangle[gold](A,B)
  \tkzGetPoint{C}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(A,B)
  \tkzLabelPoints[above](C)
 \tkzLabelAngle[pos=0.8](B,A,C){$36^\circ$}
 \tkzLabelAngle[pos=0.8](C,B,A){$72^\circ$}
 \tkzLabelAngle[pos=0.8](A,C,B){$72^\circ$}
\end{tikzpicture}
```

15.1.7. Option euclid

Euclid and **gold** are identical but the segment AB is a base in one and a side in the other.




```
\begin{tikzpicture}[scale=.75]
 \tkzDefPoint(0,0){A} \tkzDefPoint(4,0){B}
 \tkzDefTriangle[euclid](A,B)\tkzGetPoint{C}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawPoints(A,B,C)
 \tkzLabelPoints(A,B)
 \tkzLabelPoints[above](C)
 \tkzLabelAngle[pos=0.8](B,A,C){$72^\circ$}
 \tkzLabelAngle[pos=0.8](C,B,A){$72^\circ$}
 \tkzLabelAngle[pos=0.8](A,C,B){$36^\circ$}
\end{tikzpicture}
```

16. Specific triangles with \tkzDefSpcTriangle

The centers of some triangles have been defined in the "points" section, here it is a question of determining the three vertices of specific triangles.

\tkzDefSpcTriangle[⟨local options⟩](⟨p1,p2,p3⟩){⟨r1,r2,r3⟩}

The order of the points is important! p1p2p3 defines a triangle then the result is a triangle whose vertices have as reference a combination with **name** and r1,r2, r3. If **name** is empty then the references are r1,r2 and r3.

options	default	definition
orthic	centroid	determined by endpoints of the altitudes ...
centroid or medial	centroid	intersection of the triangle's three triangle medians
in or incentral	centroid	determined with the angle bisectors
ex or excentral	centroid	determined with the excenters
extouch	centroid	formed by the points of tangency with the excircles
intouch or contact	centroid	formed by the points of tangency of the incircle each of the vertices
euler	centroid	formed by Euler points on the nine-point circle
symmedial	centroid	intersection points of the symmedians
tangential	centroid	formed by the lines tangent to the circumcircle
feuerbach	centroid	formed by the points of tangency of the nine-point ... circle with the excircles
name	empty	used to name the vertices

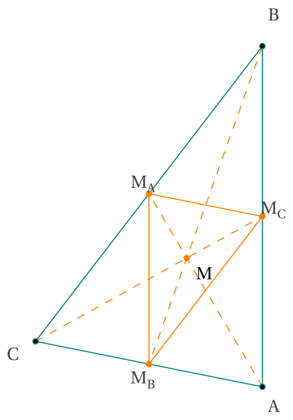
16.1. How to name the vertices

With `\tkzDefSpcTriangle[medial,name=M](A,B,C){_A,_B,_C}` you get three vertices named M_A, M_B and M_C.
With `\tkzDefSpcTriangle[medial](A,B,C){a,b,c}` you get three vertices named and labeled a, b and c.
Possible `\tkzDefSpcTriangle[medial,name=M_](A,B,C){A,B,C}` you get three vertices named M_A, M_B and M_C.

16.2. Option medial or centroid

The geometric centroid of the polygon vertices of a triangle is the point G (sometimes also denoted M) which is also the intersection of the triangle's three triangle medians. The point is therefore sometimes called the median point. The centroid is always in the interior of the triangle.

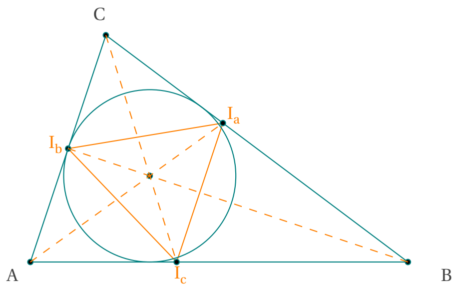
Weisstein, Eric W. "Centroid triangle" From MathWorld–A Wolfram Web Resource.
In the following example, we obtain the Euler circle which passes through the previously defined points.




```
\begin{tikzpicture}[rotate=90,scale=.75]
 \tkzDefPoints{0/0/A,6/0/B,0.8/4/C}
 \tkzDefTriangleCenter[centroid](A,B,C)
 \tkzGetPoint{M}
 \tkzDefSpcTriangle[medial,name=M](A,B,C){_A,_B,_C}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawSegments[dashed,new](A,M_A B,M_B C,M_C)
 \tkzDrawPolygon[new](M_A,M_B,M_C)
 \tkzDrawPoints(A,B,C)
 \tkzDrawPoints[new](M,M_A,M_B,M_C)
 \tkzAutoLabelPoints[center=M,font=\scriptsize]%
(A,B,C,M_A,M_B,M_C)
 \tkzLabelPoints[font=\scriptsize](M)
\end{tikzpicture}
```

16.3. Option in or incentral

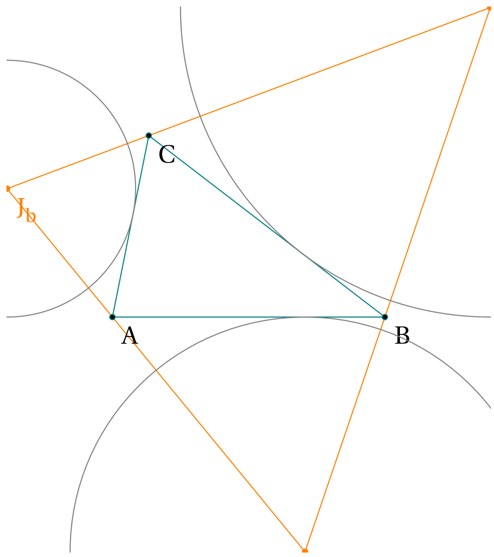
The incentral triangle is the triangle whose vertices are determined by the intersections of the reference triangle's angle bisectors with the respective opposite sides.
Weisstein, Eric W. "Incentral triangle" From MathWorld–A Wolfram Web Resource.




```
\begin{tikzpicture}[scale=1]
  \tkzDefPoints{ 0/0/A,5/0/B,1/3/C}
  \tkzDefSpcTriangle[in,name=I](A,B,C){_a,_b,_c}
  \tkzInCenter(A,B,C)\tkzGetPoint{I}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawPolygon[new](I_a,I_b,I_c)
  \tkzDrawPoints(A,B,C,I,I_a,I_b,I_c)
  \tkzDrawCircle[in](A,B,C)
  \tkzDrawSegments[dashed,new](A,I_a B,I_b C,I_c)
  \tkzAutoLabelPoints[center=I,%
  new,font=\scriptsize](I_a,I_b,I_c)
  \tkzAutoLabelPoints[center=I,
    font=\scriptsize](A,B,C)
\end{tikzpicture}
```

16.4. Option ex or excentral

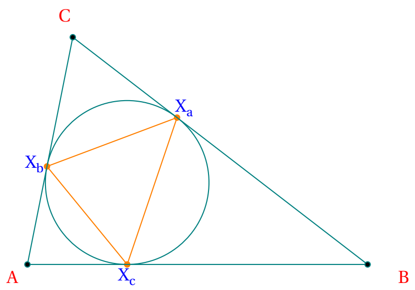
The excentral triangle of a triangle ABC is the triangle $J_aJ_bJ_c$ with vertices corresponding to the excenters of ABC.




```
\begin{tikzpicture}[scale=.6]
 \tkzDefPoints{0/0/A,6/0/B,0.8/4/C}
 \tkzDefSpcTriangle[excentral,name=J](A,B,C){_a,_b,_c}
 \tkzDefSpcTriangle[extouch,name=T](A,B,C){_a,_b,_c}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawPolygon[new](J_a,J_b,J_c)
 \tkzClipBB
 \tkzDrawPoints(A,B,C)
 \tkzDrawPoints[new](J_a,J_b,J_c)
 \tkzLabelPoints(A,B,C)
 \tkzLabelPoints[new](J_b,J_c)
 \tkzLabelPoints[new,above](J_a)
 \tkzDrawCircles[gray](J_a,T_a J_b,T_b J_c,T_c)
\end{tikzpicture}
```

16.5. Option intouch or contact

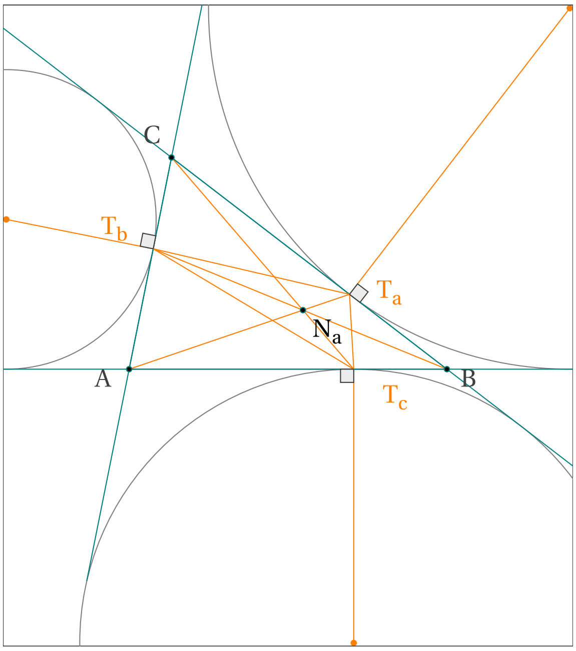
The contact triangle of a triangle ABC, also called the intouch triangle, is the triangle formed by the points of tangency of the incircle of ABC with ABC.
Weisstein, Eric W. "Contact triangle" From MathWorld–A Wolfram Web Resource.
We obtain the intersections of the bisectors with the sides.




```
\begin{tikzpicture}[scale=.75]
 \tkzDefPoints{0/0/A,6/0/B,0.8/4/C}
 \tkzDefSpcTriangle[intouch,name=X](A,B,C){_a,_b,_c}
 \tkzInCenter(A,B,C)\tkzGetPoint{I}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawPolygon[new](X_a,X_b,X_c)
 \tkzDrawPoints(A,B,C)
 \tkzDrawPoints[new](X_a,X_b,X_c)
 \tkzDrawCircle[in](A,B,C)
 \tkzAutoLabelPoints[center=I,blue,font=\scriptsize]%
(X_a,X_b,X_c)
 \tkzAutoLabelPoints[center=I,red,font=\scriptsize]%
(A,B,C)
\end{tikzpicture}
```

16.6. Option extouch

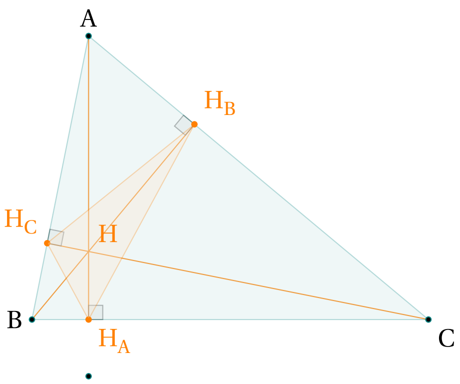
The extouch triangle $T_aT_bT_c$ is the triangle formed by the points of tangency of a triangle ABC with its excircles J_a, J_b, and J_c. The points T_a, T_b, and T_c can also be constructed as the points which bisect the perimeter of $A_1A_2A_3$ starting at A, B, and C.
Weisstein, Eric W. "Extouch triangle" From MathWorld–A Wolfram Web Resource.
We obtain the points of contact of the exinscribed circles as well as the triangle formed by the centres of the exinscribed circles.




```
\begin{tikzpicture}[scale=.7]
\tkzDefPoints{0/0/A,6/0/B,0.8/4/C}
\tkzDefSpcTriangle[excentral,
                  name=J](A,B,C){_a,_b,_c}
\tkzDefSpcTriangle[extouch,
                   name=T](A,B,C){_a,_b,_c}
\tkzDefTriangleCenter[nagel](A,B,C)
\tkzGetPoint{N_a}
\tkzDefTriangleCenter[centroid](A,B,C)
\tkzGetPoint{G}
\tkzDrawPoints[new](J_a,J_b,J_c)
\tkzClipBB \tkzShowBB
\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,N_a)
\tkzLabelPoints(N_a)
\tkzAutoLabelPoints[center=N_a](A,B,C)
\tkzAutoLabelPoints[center=G,new,
                           dist=.4](T_a,T_b,T_c)
\tkzMarkRightAngles[fill=gray!15](J_a,T_a,B
 J_b,T_b,C J_c,T_c,A)
\end{tikzpicture}
```

16.7. Option orthic

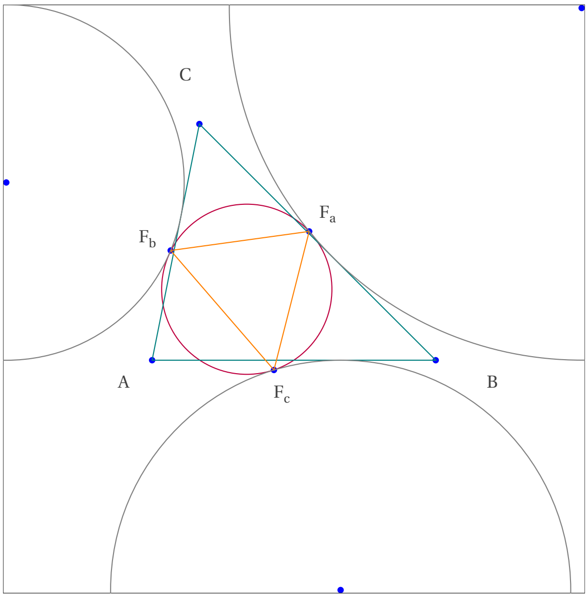
Given a triangle ABC, the triangle $H_AH_BH_C$ whose vertices are endpoints of the altitudes from each of the vertices of ABC is called the orthic triangle, or sometimes the altitude triangle. The three lines AH_A, BH_B, and CH_C are concurrent at the orthocenter H of ABC.




```
\begin{tikzpicture}[scale=.75]
\tkzDefPoints{1/5/A,0/0/B,7/0/C}
 \tkzDefSpcTriangle[orthic](A,B,C){H_A,H_B,H_C}
 \tkzDefTriangleCenter[ortho](B,C,A)
 \tkzGetPoint{H}
 \tkzDefPointWith[orthogonal,normed](H_A,B)
 \tkzGetPoint{a}
 \tkzDrawSegments[new](A,H_A B,H_B C,H_C)
 \tkzMarkRightAngles[fill=gray!20,
        opacity=.5](A,H_A,C B,H_B,A C,H_C,A)
 \tkzDrawPolygon[fill=teal!20,opacity=.3](A,B,C)
 \tkzDrawPoints(A,B,C)
 \tkzDrawPoints[new](H_A,H_B,H_C)
 \tkzDrawPolygon[new,fill=orange!20,
               opacity=.3](H_A,H_B,H_C)
 \tkzDrawPoint(a)
 \tkzLabelPoints(C)
 \tkzLabelPoints[left](B)
 \tkzLabelPoints[above](A)
 \tkzLabelPoints[new](H_A)
 \tkzLabelPoints[new,above left](H_C)
 \tkzLabelPoints[new,above right](H_B,H)
\end{tikzpicture}
```

16.8. Option feuerbach

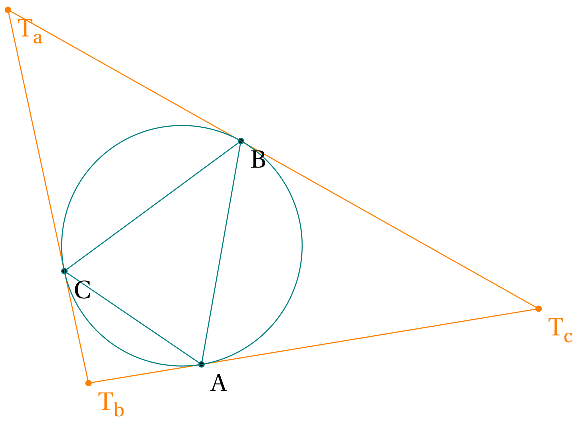
The Feuerbach triangle is the triangle formed by the three points of tangency of the nine-point circle with the excircles.
Weisstein, Eric W. "Feuerbach triangle" From MathWorld–A Wolfram Web Resource.
The points of tangency define the Feuerbach triangle.




```
\begin{tikzpicture}[scale=1.25]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(3,0){B}
  \tkzDefPoint(0.5,2.5){C}
  \tkzDefCircle[euler](A,B,C) \tkzGetPoint{N}
  \tkzDefSpcTriangle[feuerbach,
                           name=F](A,B,C){_a,_b,_c}
  \tkzDefSpcTriangle[excentral,
                           name=J](A,B,C){_a,_b,_c}
  \tkzDefSpcTriangle[extouch,
                            name=T](A,B,C){_a,_b,_c}
  \tkzDrawPoints[blue](J_a,J_b,J_c,%
          F_a,F_b,F_c,A,B,C)
  \tkzClipBB \tkzShowBB
  \tkzDrawCircle[purple](N,F_a)
  \tkzDrawPolygon(A,B,C)
  \tkzDrawPolygon[new](F_a,F_b,F_c)
  \tkzDrawCircles[gray](J_a,F_a J_b,F_b J_c,F_c)
  \tkzAutoLabelPoints[center=N,dist=.3,
   font=\scriptsize](A,B,C,F_a,F_b,%
                     F_c,J_a,J_b,J_c)
\end{tikzpicture}
```

16.9. Option tangential

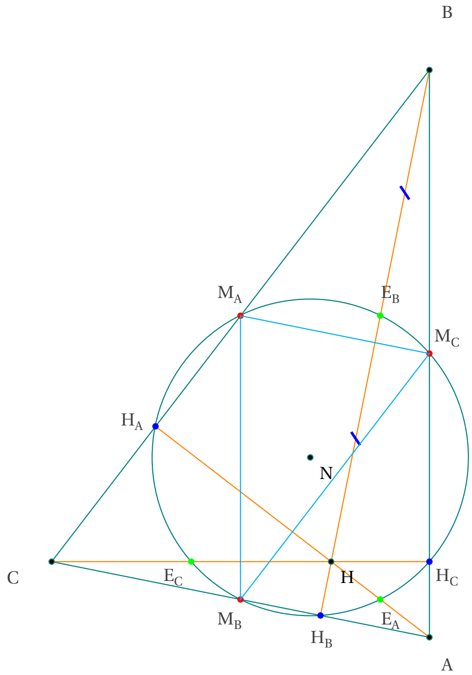
The tangential triangle is the triangle $T_aT_bT_c$ formed by the lines tangent to the circumcircle of a given triangle ABC at its vertices. It is therefore antipedal triangle of ABC with respect to the circumcenter O.
Weisstein, Eric W. "Tangential Triangle." From MathWorld–A Wolfram Web Resource.




```
\begin{tikzpicture}[scale=.5,rotate=80]
  \tkzDefPoints{0/0/A,6/0/B,1.8/4/C}
  \tkzDefSpcTriangle[tangential,
    name=T](A,B,C){_a,_b,_c}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawPolygon[new](T_a,T_b,T_c)
  \tkzDrawPoints(A,B,C)
  \tkzDrawPoints[new](T_a,T_b,T_c)
  \tkzDefCircle[circum](A,B,C)
  \tkzGetPoint{O}
  \tkzDrawCircle(O,A)
  \tkzLabelPoints(A,B,C)
  \tkzLabelPoints[new](T_a,T_b,T_c)
\end{tikzpicture}
```

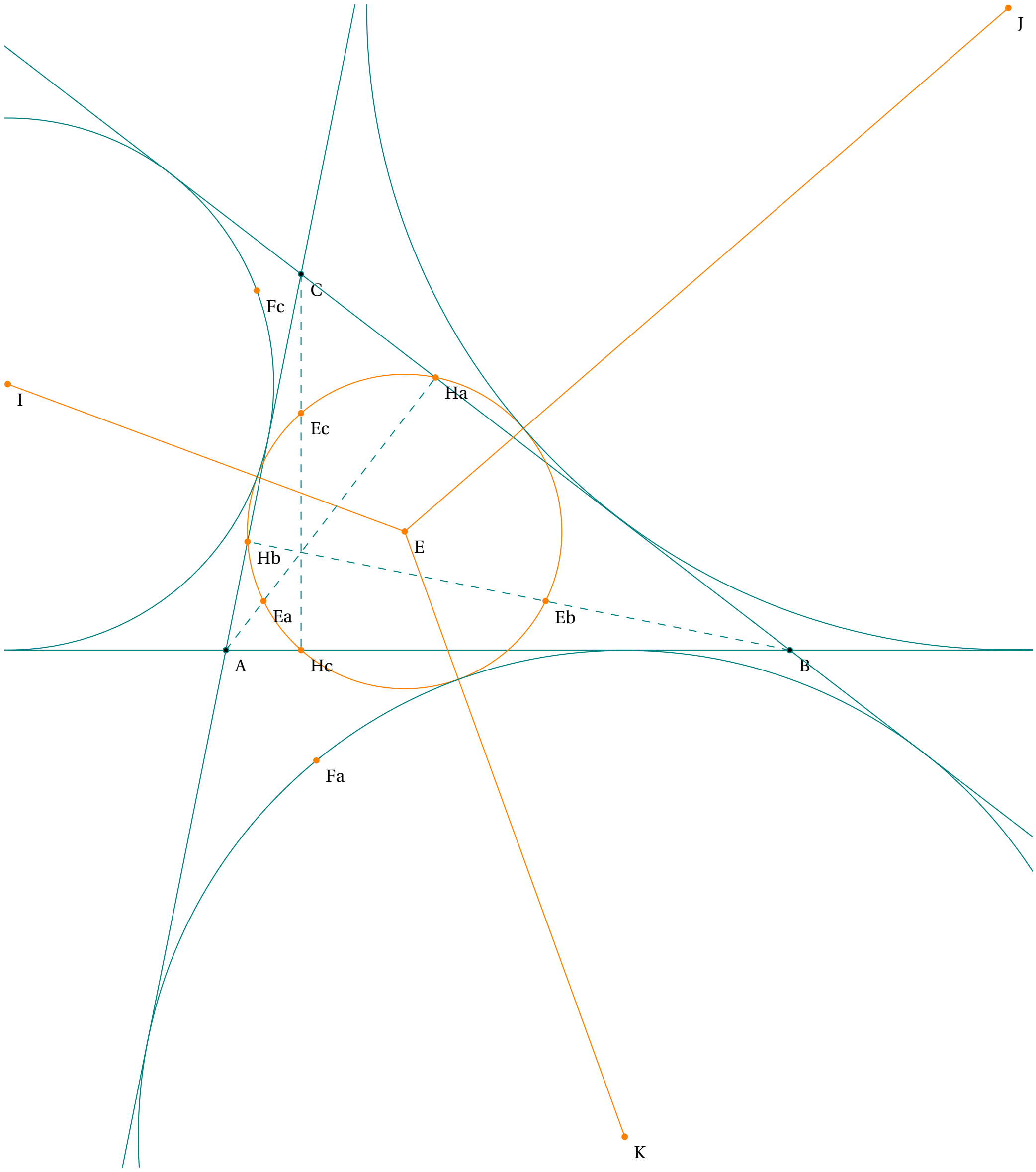
16.10. Option euler

The Euler triangle of a triangle ABC is the triangle $E_AE_BE_C$ whose vertices are the midpoints of the segments joining the orthocenter H with the respective vertices. The vertices of the triangle are known as the Euler points, and lie on the nine-point circle.
Weisstein, Eric W. "Euler Triangle." From MathWorld–A Wolfram Web Resource.




```
\begin{tikzpicture}[rotate=90,scale=1.25]
 \tkzDefPoints{0/0/A,6/0/B,0.8/4/C}
 \tkzDefSpcTriangle[medial,
     name=M](A,B,C){_A,_B,_C}
 \tkzDefTriangleCenter[euler](A,B,C)
     \tkzGetPoint{N} % I= N nine points
 \tkzDefTriangleCenter[ortho](A,B,C)
        \tkzGetPoint{H}
 \tkzDefMidPoint(A,H) \tkzGetPoint{E_A}
 \tkzDefMidPoint(C,H) \tkzGetPoint{E_C}
 \tkzDefMidPoint(B,H) \tkzGetPoint{E_B}
 \tkzDefSpcTriangle[ortho,name=H](A,B,C){_A,_B,_C}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawCircle(N,E_A)
 \tkzDrawSegments[new](A,H_A B,H_B C,H_C)
 \tkzDrawPoints(A,B,C,N,H)
 \tkzDrawPoints[red](M_A,M_B,M_C)
 \tkzDrawPoints[blue]( H_A,H_B,H_C)
 \tkzDrawPoints[green](E_A,E_B,E_C)
 \tkzAutoLabelPoints[center=N,font=\scriptsize]%
(A,B,C,M_A,M_B,M_C,H_A,H_B,H_C,E_A,E_B,E_C)
\tkzLabelPoints[font=\scriptsize](H,N)
\tkzMarkSegments[mark=s|,size=3pt,
  color=blue,line width=1pt](B,E_B E_B,H)
   \tkzDrawPolygon[color=cyan](M_A,M_B,M_C)
\end{tikzpicture}
```

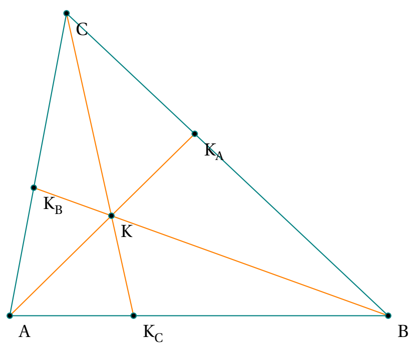
16.11. Option euler and Option orthic



```
\begin{tikzpicture}[scale=1.25]
  \tkzDefPoints{0/0/A,6/0/B,0.8/4/C}
  \tkzDefSpcTriangle[euler,name=E](A,B,C){a,b,c}
  \tkzDefSpcTriangle[orthic,name=H](A,B,C){a,b,c}
  \tkzDefExCircle(A,B,C) \tkzGetPoint{I} \tkzGetLength{rI}
  \tkzDefExCircle(C,A,B) \tkzGetPoint{J} \tkzGetLength{rJ}
  \tkzDefExCircle(B,C,A) \tkzGetPoint{K} \tkzGetLength{rK}
  \tkzDrawPoints[orange](I,J,K)
  \tkzLabelPoints[font=\scriptsize](A,B,C,I,J,K)
  \tkzClipBB
  \tkzInterLC[R](I,C)(I,\rI) \tkzGetSecondPoint{Fc}
  \tkzInterLC[R](J,B)(J,\rJ) \tkzGetSecondPoint{Fb}
  \tkzInterLC[R](K,A)(K,\rK) \tkzGetSecondPoint{Fa}
  \tkzDrawLines[add=1.5 and 1.5](A,B A,C B,C)
  \tkzDrawCircle[euler,orange](A,B,C) \tkzGetPoint{E}
  \tkzDrawSegments[orange](E,I E,J E,K)
  \tkzDrawSegments[dashed](A,Ha B,Hb C,Hc)
  \tkzDrawCircles[R](J,{\rJ} I,{\rI} K,{\rK})
  \tkzDrawPoints(A,B,C)
  \tkzDrawPoints[orange](E,I,J,K,Ha,Hb,Hc,Ea,Eb,Ec,Fa,Fb,Fc)
  \tkzLabelPoints[font=\scriptsize](E,Ea,Eb,Ec,Ha,Hb,Hc,Fa,Fb,Fc)
\end{tikzpicture}
```

16.12. Option symmedial

The symmedial triangle$K_AK_BK_C$ is the triangle whose vertices are the intersection points of the symmedians with the reference triangle ABC.




```
\begin{tikzpicture}
\tkzDefPoint(0,0){A}
\tkzDefPoint(5,0){B}
\tkzDefPoint(.75,4){C}
\tkzDefTriangleCenter[symmedian](A,B,C)\tkzGetPoint{K}
\tkzDefSpcTriangle[symmedial,name=K_](A,B,C){A,B,C}
\tkzDrawPolygon(A,B,C)
\tkzDrawSegments[new](A,K_A B,K_B C,K_C)
\tkzDrawPoints(A,B,C,K,K_A,K_B,K_C)
\tkzLabelPoints[font=\scriptsize](A,B,C,K,K_A,K_B,K_C)
\end{tikzpicture}
```

17. Definition of polygons

17.1. Defining the points of a square

We have seen the definitions of some triangles. Let us look at the definitions of some quadrilaterals and regular polygons.

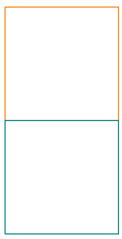
\tkzDefSquare(⟨pt1,pt2⟩)

The square is defined in the forward direction. From two points, two more points are obtained such that the four taken in order form a square. The square is defined in the forward direction.
The results are in `tkzFirstPointResult` and `tkzSecondPointResult`.
We can rename them with `\tkzGetPoints`.

Arguments	example	explication
(⟨pt1,pt2⟩)	\tkzDefSquare(⟨A,B⟩)	The square is defined in the direct direction.

17.1.1. Using \tkzDefSquare with two points

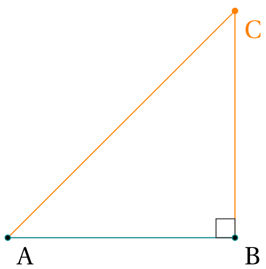
Note the inversion of the first two points and the result.



```
\begin{tikzpicture}[scale=.5]
  \tkzDefPoint(0,0){A} \tkzDefPoint(3,0){B}
  \tkzDefSquare(A,B)
  \tkzDrawPolygon[new](A,B,tkzFirstPointResult,%
             tkzSecondPointResult)
  \tkzDefSquare(B,A)
  \tkzDrawPolygon(B,A,tkzFirstPointResult,%
             tkzSecondPointResult)
\end{tikzpicture}
```

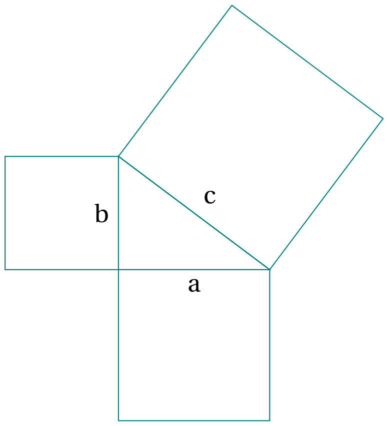
We may only need one point to draw an isosceles right-angled triangle so we use `\tkzGetFirstPoint` or `\tkzGetSecondPoint`.

17.1.2. Use of \tkzDefSquare to obtain an isosceles right-angled triangle




```
\begin{tikzpicture}[scale=1]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(3,0){B}
  \tkzDefSquare(A,B) \tkzGetFirstPoint{C}
  \tkzDrawSegment(A,B)
  \tkzDrawSegments[new](A,C B,C)
  \tkzMarkRightAngles(A,B,C)
  \tkzDrawPoints(A,B) \tkzDrawPoint[new](C)
  \tkzLabelPoints(A,B)
  \tkzLabelPoints[new](C)
\end{tikzpicture}
```

17.1.3. Pythagorean Theorem and \tkzDefSquare




```
\begin{tikzpicture}[scale=.5]
\tkzDefPoint(0,0){C}
\tkzDefPoint(4,0){A}
\tkzDefPoint(0,3){B}
\tkzDefSquare(B,A)\tkzGetPoints{E}{F}
\tkzDefSquare(A,C)\tkzGetPoints{G}{H}
\tkzDefSquare(C,B)\tkzGetPoints{I}{J}
\tkzDrawPolygon(A,B,C)
\tkzDrawPolygon(A,C,G,H)
\tkzDrawPolygon(C,B,I,J)
\tkzDrawPolygon(B,A,E,F)
\tkzLabelSegment(A,C){$a$}
\tkzLabelSegment(C,B){$b$}
\tkzLabelSegment[swap](A,B){$c$}
\end{tikzpicture}
```

17.2. Defining the points of a rectangle

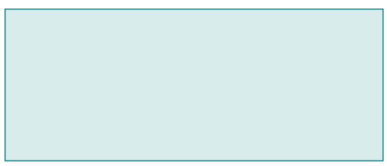
.

\tkzDefRectangle(⟨pt1,pt2⟩)

The rectangle is defined in the forward direction. From two points, two more points are obtained such that the four taken in order form a rectangle. The two points passed in arguments are the ends of a diagonal of the rectangle. The sides are parallel to the axes.
The results are in **tkzFirstPointResult** and **tkzSecondPointResult**.
We can rename them with **\tkzGetPoints**.

Arguments	example	explication
(⟨pt1,pt2⟩)	\tkzDefRectangle(⟨A,B⟩)	The rectangle is defined in the direct direction.

17.2.1. Example of a rectangle definition



```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,5/2/C}
\tkzDefRectangle(A,C) \tkzGetPoints{B}{D}
\tkzDrawPolygon[fill=teal!15](A,...,D)
\end{tikzpicture}
```

17.3. Definition of parallelogram

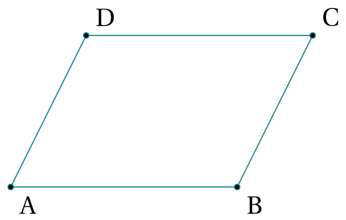
Defining the points of a parallelogram. It is a matter of completing three points in order to obtain a parallelogram.

\tkzDefParallelogram(⟨pt1,pt2,pt3⟩)

arguments	default	definition
(⟨pt1,pt2,pt3⟩)	no default	Three points are necessary

From three points, another point is obtained such that the four taken in order form a parallelogram.
The result is in **tkzPointResult**.
We can rename it with the name `\tkzGetPoint`...

17.3.1. Example of a parallelogram definition




```
\begin{tikzpicture}[scale=1]
 \tkzDefPoints{0/0/A,3/0/B,4/2/C}
 \tkzDefParallelogram(A,B,C)
 % or   \tkzDefPointWith[colinear= at C](B,A)
 \tkzGetPoint{D}
 \tkzDrawPolygon(A,B,C,D)
 \tkzLabelPoints(A,B)
 \tkzLabelPoints[above right](C,D)
 \tkzDrawPoints(A,...,D)
\end{tikzpicture}
```

17.4. The golden rectangle

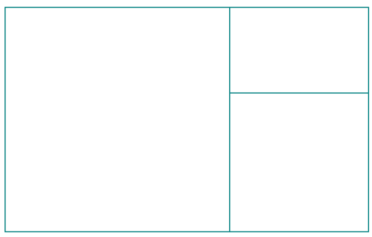
`\tkzDefGoldenRectangle(⟨point,point⟩)`

The macro determines a rectangle whose size ratio is the number Φ.
The created points are in **tkzFirstPointResult** and **tkzSecondPointResult**.
They can be obtained with the macro `\tkzGetPoints`. The following macro is used to draw the rectangle.

arguments	example	explication
(⟨pt1,pt2⟩)	(⟨A,B⟩)	If C and D are created then $AB/BC = \Phi$.

`\tkzDefGoldenRectangle` or `\tkzDefGoldRectangle`

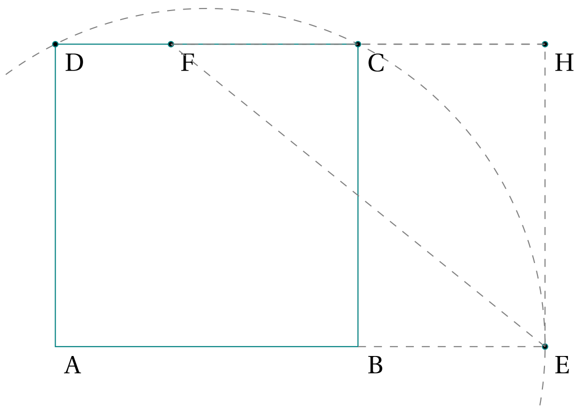
17.4.1. Golden Rectangles



```
\begin{tikzpicture}[scale=.6]
 \tkzDefPoint(0,0){A}      \tkzDefPoint(8,0){B}
 \tkzDefGoldRectangle(A,B) \tkzGetPoints{C}{D}
 \tkzDefGoldRectangle(B,C) \tkzGetPoints{E}{F}
 \tkzDefGoldRectangle(C,E) \tkzGetPoints{G}{H}
 \tkzDrawPolygon(A,B,C,D)
 \tkzDrawSegments(E,F G,H)
\end{tikzpicture}
```

17.4.2. Construction of the golden rectangle

Without the previous macro here is how to get the golden rectangle.




```
\begin{tikzpicture}[scale=.5]
\tkzDefPoint(0,0){A}
\tkzDefPoint(8,0){B}
\tkzDefMidPoint(A,B)
\tkzGetPoint{I}
\tkzDefSquare(A,B)\tkzGetPoints{C}{D}
\tkzInterLC(A,B)(I,C)\tkzGetPoints{G}{E}
\tkzDefPointWith[colinear= at C](E,B)
 \tkzGetPoint{F}
\tkzDefPointBy[projection=onto D--C ](E)
 \tkzGetPoint{H}
\tkzDrawArc[style=dashed](I,E)(D)
\tkzDrawSquare(A,B)
\tkzDrawPoints(C,D,E,F,H)
\tkzLabelPoints(A,B,C,D,E,F,H)
\tkzDrawSegments[style=dashed,color=gray]%
(E,F C,F B,E F,H H,C E,H)
\end{tikzpicture}
```

17.5. Regular polygon

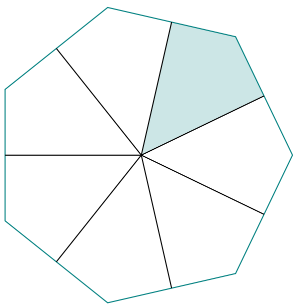
`\tkzDefRegPolygon[⟨local options⟩](⟨pt1,pt2⟩)`

From the number of sides, depending on the options, this macro determines a regular polygon according to its center or one side.

arguments	example	explication
(⟨pt1,pt2⟩)	(⟨O,A⟩)	with option "center", O is the center of the polygon.
(⟨pt1,pt2⟩)	(⟨A,B⟩)	with option "side", [AB] is a side.

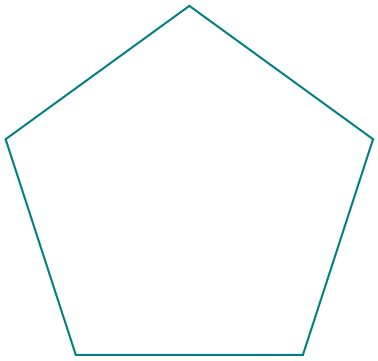
options	default	example
name	P	The vertices are named P1,P2,…
sides	5	number of sides.
center	center	The first point is the center.
side	center	The two points are vertices.
Options TikZ	...	

17.5.1. Option center



```
\begin{tikzpicture}
  \tkzDefPoints{0/0/P0,0/0/Q0,2/0/P1}
  \tkzDefMidPoint(P0,P1) \tkzGetPoint{Q1}
  \tkzDefRegPolygon[center,sides=7](P0,P1)
  \tkzDefMidPoint(P1,P2) \tkzGetPoint{Q1}
  \tkzDefRegPolygon[center,sides=7,name=Q](P0,Q1)
  \tkzFillPolygon[teal!20](Q0,Q1,P2,Q2)
  \tkzDrawPolygon(P1,P...,P7)
  \foreach \j in {1,...,7} {%
  \tkzDrawSegment[black](P0,Q\j)}
\end{tikzpicture}
```

17.5.2. Option side



```
\begin{tikzpicture}[scale=1]
    \tkzDefPoints{-4/0/A, -1/0/B}
    \tkzDefRegPolygon[side,sides=5,name=P](A,B)
    \tkzDrawPolygon[thick](P1,P...,P5)
\end{tikzpicture}
```

18. The Circles

Among the following macros, one will allow you to draw a circle, which is not a real feat. To do this, you will need to know the center of the circle and either the radius of the circle or a point on the circumference. It seemed to me that the most frequent use was to draw a circle with a given centre passing through a given point. This will be the default method, otherwise you will have to use the **R** option. There are a large number of special circles, for example the circle circumscribed by a triangle.

- I have created a first macro `\tkzDefCircle` which allows, according to a particular circle, to retrieve its center and the measurement of the radius in cm. This recovery is done with the macros `\tkzGetPoint` and `\tkzGetLength`;
- then a macro `\tkzDrawCircle`;
- then a macro that allows you to color in a disc, but without drawing the circle `\tkzFillCircle`;
- sometimes, it is necessary for a drawing to be contained in a disk, this is the role assigned to `\tkzClipCircle`;
- it finally remains to be able to give a label to designate a circle and if several possibilities are offered, we will see here `\tkzLabelCircle`.

18.1. Characteristics of a circle: \tkzDefCircle

This macro allows you to retrieve the characteristics (center and radius) of certain circles.

```
\tkzDefCircle[⟨local options⟩](⟨A,B⟩) or (⟨A,B,C⟩)
```

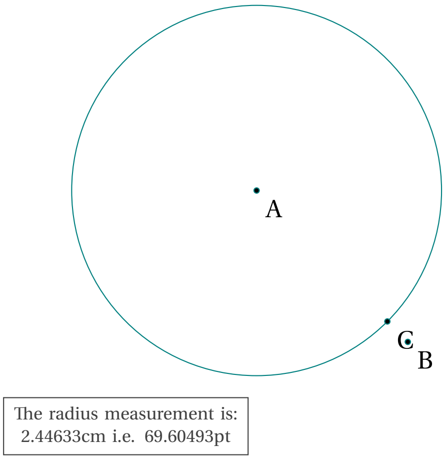
Attention the arguments are lists of two or three points. This macro is either used in partnership with `\tkzGetPoint` and/or `\tkzGetLength` to obtain the center and the radius of the circle, or by using **tkzPointResult** and **tkzLengthResult** if it is not necessary to keep the results.

arguments	example	explication
(⟨pt1,pt2⟩) or (⟨pt1,pt2,pt3⟩)	(⟨A,B⟩)	[AB] is radius A is the center

options	default	definition
through	through	circle characterized by two points defining a radius
diameter	through	circle characterized by two points defining a diameter
circum	through	circle circumscribed of a triangle
in	through	incircle a triangle
ex	through	excircle of a triangle
euler or nine	through	Euler's Circle
spieker	through	Spieker Circle
apollonius	through	circle of Apollonius
K	1	coefficient used for a circle of Apollonius

In the following examples, I draw the circles with a macro not yet presented, but this is not necessary. In some cases you may only need the center or the radius.

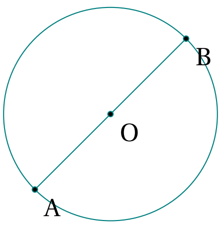
18.1.1. Example with a random point and option through




```
\begin{tikzpicture}[scale=1]
  \tkzDefPoint(0,4){A}
  \tkzDefPoint(2,2){B}
  \tkzDefMidPoint(A,B) \tkzGetPoint{I}
  \tkzDefRandPointOn[segment = I--B]
  \tkzGetPoint{C}
  \tkzDefCircle[through](A,C)
  \tkzGetLength{rACcm}
  \tkzcmtopt(\rACcm){rACpt}
  \tkzDrawCircle(A,C)
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(A,B,C)
  \tkzLabelCircle[draw,
           text width=3cm,text centered,
           font=\scriptsize,below=1cm](A,C)(-90)%
  {The radius measurement is:
   \rACcm cm i.e. \rACpt pt}
\end{tikzpicture}
```

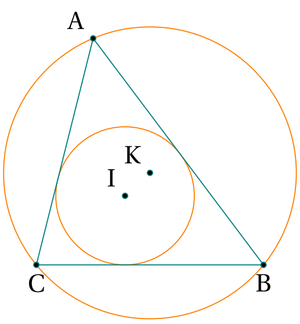
18.1.2. Example with option diameter

It is simpler here to search directly for the middle of [AB].




```
\begin{tikzpicture}[scale=1]
   \tkzDefPoint(0,0){A}
   \tkzDefPoint(2,2){B}
   \tkzDefCircle[diameter](A,B)
   \tkzGetPoint{O}
   \tkzDrawCircle(O,B)
   \tkzDrawSegment(A,B)
   \tkzDrawPoints(A,B,O)
   \tkzLabelPoints(A,B,O)
\end{tikzpicture}
```

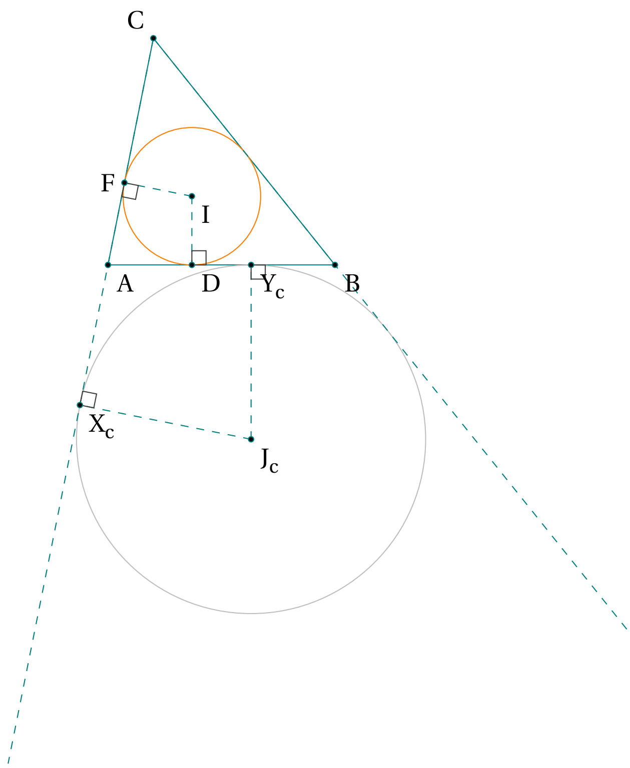
18.1.3. Circles inscribed and circumscribed for a given triangle




```
\begin{tikzpicture}[scale=.75]
 \tkzDefPoint(2,2){A}  \tkzDefPoint(5,-2){B}
 \tkzDefPoint(1,-2){C}
 \tkzDefCircle[in](A,B,C)
 \tkzGetPoint{I}    \tkzGetLength{rIN}
 \tkzDefCircle[circum](A,B,C)
 \tkzGetPoint{K}   \tkzGetLength{rCI}
 \tkzDrawCircles[R,new](I,{\rIN} K,{\rCI})
 \tkzLabelPoints[below](B,C)
 \tkzLabelPoints[above left](A,I,K)
 \tkzDrawPolygon(A,B,C)
 \tkzDrawPoints(A,B,C,I,K)
\end{tikzpicture}
```

18.1.4. Example with option ex

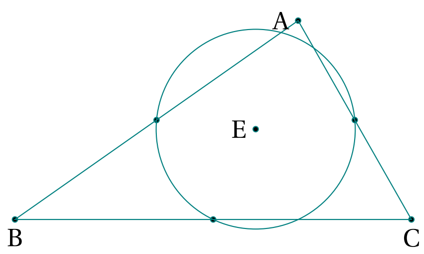
We want to define an excircle of a triangle relatively to point C




```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoints{ 0/0/A,4/0/B,0.8/4/C}
  \tkzDefCircle[ex](B,C,A)
  \tkzGetPoint{J_c} \tkzGetLength{rc}
  \tkzDefPointBy[projection=onto A--C ](J_c)
  \tkzGetPoint{X_c}
  \tkzDefPointBy[projection=onto A--B ](J_c)
  \tkzGetPoint{Y_c}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawCircle[R,color=lightgray](J_c,\rc)
  % possible  \tkzDrawCircle[ex](A,B,C)
  \tkzDrawCircle[in,new](A,B,C)
  \tkzGetPoint{I}
  \tkzDefPointBy[projection=onto A--C ](I)
  \tkzGetPoint{F}
  \tkzDefPointBy[projection=onto A--B ](I)
  \tkzGetPoint{D}
  \tkzDrawLines[add=0 and 2.2,dashed](C,A C,B)
  \tkzDrawSegments[dashed](J_c,X_c I,D  I,F%
          J_c,Y_c)
  \tkzMarkRightAngles(A,F,I B,D,I J_c,X_c,A%
         J_c,Y_c,B)
  \tkzDrawPoints(B,C,A,I,D,F,X_c,J_c,Y_c)
  \tkzLabelPoints(B,A,J_c,I,D,X_c,Y_c)
  \tkzLabelPoints[above left](C)
  \tkzLabelPoints[left](F)
\end{tikzpicture}
```

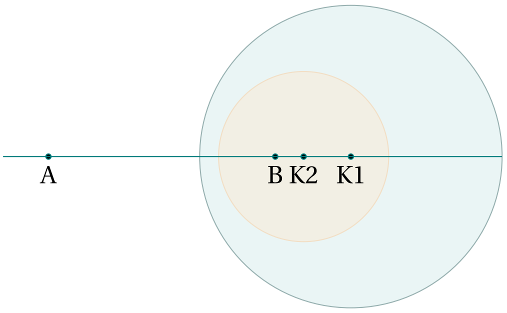
18.1.5. Euler's circle for a given triangle with option `euler`

We verify that this circle passes through the middle of each side.




```
\begin{tikzpicture}[scale=.75]
   \tkzDefPoint(5,3.5){A}
   \tkzDefPoint(0,0){B} \tkzDefPoint(7,0){C}
   \tkzDefCircle[euler](A,B,C)
   \tkzGetPoint{E}  \tkzGetLength{rEuler}
   \tkzDefSpcTriangle[medial](A,B,C){M_a,M_b,M_c}
   \tkzDrawPoints(A,B,C,E,M_a,M_b,M_c)
   \tkzDrawCircle[R](E,\rEuler)
   \tkzDrawPolygon(A,B,C)
   \tkzLabelPoints[below](B,C)
   \tkzLabelPoints[left](A,E)
\end{tikzpicture}
```

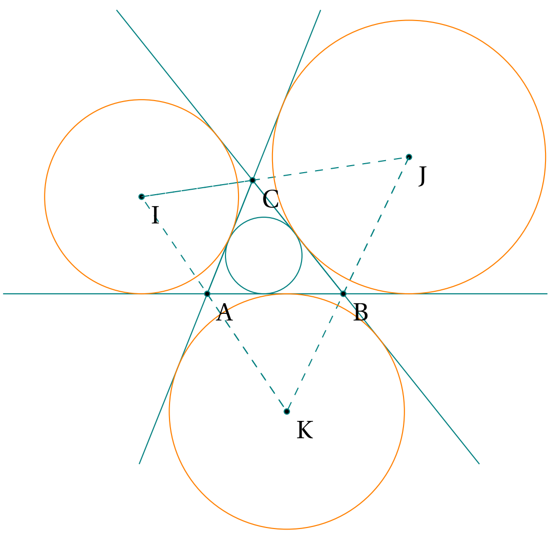
18.1.6. Apollonius circles for a given segment option apollonius




```
\begin{tikzpicture}[scale=0.75]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(4,0){B}
  \tkzDefCircle[apollonius,K=2](A,B)
  \tkzGetPoint{K1}
  \tkzGetLength{rAp}
  \tkzDrawCircle[R,color = teal!50!black,
      fill=teal!20,opacity=.4](K1,\rAp)
  \tkzDefCircle[apollonius,K=3](A,B)
  \tkzGetPoint{K2}   \tkzGetLength{rAp}
  \tkzDrawCircle[R,color=orange!50,
   fill=orange!20,opacity=.4](K2,\rAp)
  \tkzLabelPoints[below](A,B,K1,K2)
  \tkzDrawPoints(A,B,K1,K2)
  \tkzDrawLine[add=.2 and 1](A,B)
\end{tikzpicture}
```

18.1.7. Circles exinscribed to a given triangle option ex

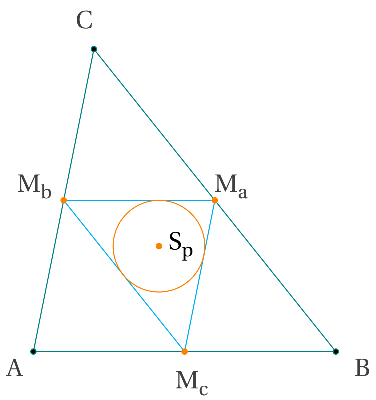
You can also get the center and the projection of it on one side of the triangle.
with `\tkzGetFirstPoint{Jb}` and `\tkzGetSecondPoint{Tb}`.




```
\begin{tikzpicture}[scale=.6]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(3,0){B}
  \tkzDefPoint(1,2.5){C}
  \tkzDefCircle[ex](A,B,C) \tkzGetPoint{I}
    \tkzGetLength{rI}
  \tkzDefCircle[ex](C,A,B) \tkzGetPoint{J}
    \tkzGetLength{rJ}
  \tkzDefCircle[ex](B,C,A) \tkzGetPoint{K}
    \tkzGetLength{rK}
   \tkzDefCircle[in](B,C,A) \tkzGetPoint{O}
     \tkzGetLength{rO}
  \tkzDrawLines[add=1.5 and 1.5](A,B A,C B,C)
  \tkzDrawPoints(I,J,K)
  \tkzDrawPolygon(A,B,C)
  \tkzDrawPolygon[dashed](I,J,K)
  \tkzDrawCircle[R,teal](O,\rO)
  \tkzDrawSegments[dashed](A,K B,J C,I)
  \tkzDrawPoints(A,B,C)
  \tkzDrawCircles[R,new](J,{\rJ} I,{\rI}%
                         K,{\rK})
  \tkzLabelPoints(A,B,C,I,J,K)
\end{tikzpicture}
```

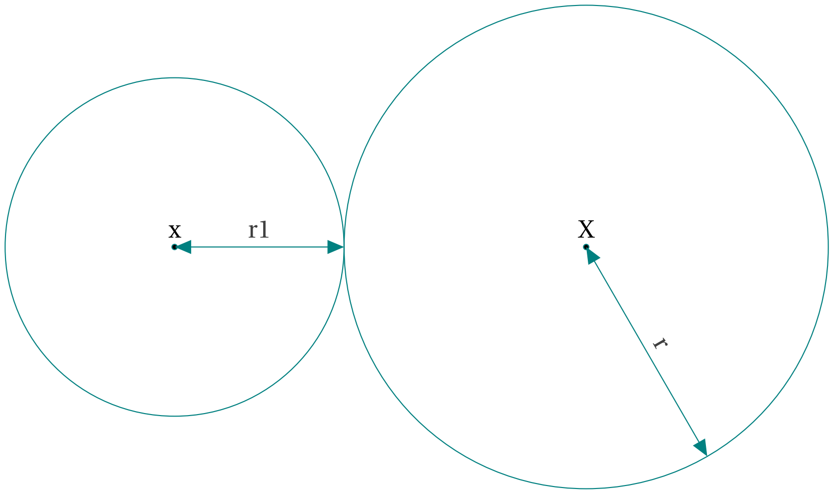
18.1.8. Spieker circle with option spieker

The incircle of the medial triangle $M_aM_bM_c$ is the Spieker circle:

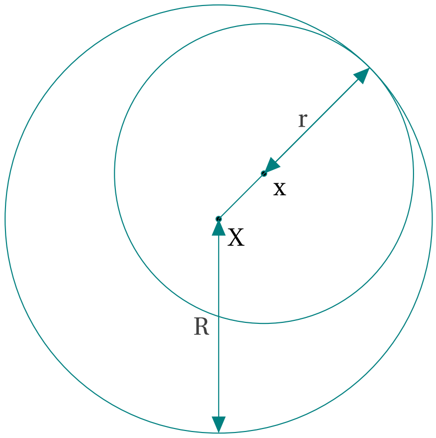



```
\begin{tikzpicture}[scale=1]
  \tkzDefPoints{ 0/0/A,4/0/B,0.8/4/C}
   \tkzDefSpcTriangle[medial](A,B,C){M_a,M_b,M_c}
   \tkzDefTriangleCenter[spieker](A,B,C)
   \tkzGetPoint{S_p}
   \tkzDrawPolygon(A,B,C)
   \tkzDrawPolygon[cyan](M_a,M_b,M_c)
   \tkzDrawPoints(B,C,A)
   \tkzDrawPoints[new](M_a,M_b,M_c,S_p)
   \tkzDrawCircle[in,new](M_a,M_b,M_c)
   \tkzAutoLabelPoints[center=S_p,dist=.3](M_a,M_b,M_c)
   \tkzLabelPoints[right](S_p)
   \tkzAutoLabelPoints[center=S_p](A,B,C)
\end{tikzpicture}
```

18.1.9. Examples from js bibra tex.stackexchange.com




```
\begin{tikzpicture}[scale=0.4]
\tkzDefPoint(6,4){A}
\tkzDefPoint(6,-4){B}
\tkzDefMidPoint(B,A)\tkzGetPoint{P}
\tkzDefLine[orthogonal =through P](A,B)
\tkzGetPoint{X}
\tkzDefCircle[through](X,P)
\tkzCalcLength(X,P)\tkzGetLength{rXP}
\tkzDefShiftPoint[X](180:\rXP*2){y}
\tkzDefPointWith[linear,K=0.3](y,P)
 \tkzGetPoint{x}
\tkzDrawPoints(X,x)
\tkzDrawCircles(x,P X,P)
\tkzLabelLine[pos=0.5,above](x,P){r1}
\tkzDefShiftPoint[X](-60:\rXP){X'}
\tkzDrawSegments[<->, >=triangle 45](X,X' P,x)
\tkzLabelLine[pos=0.5,above, sloped](X,X'){r}
\tkzLabelPoints[above](x)
\tkzLabelPoints[above](X)
\end{tikzpicture}
```




```
\begin{tikzpicture}
    \tkzDefPoint(0,4){A}
    \tkzDefPoint(2,2){B}
    \tkzDefMidPoint(B,A)\tkzGetPoint{P}
    \tkzDefLine[orthogonal =through P](B,A)
    \tkzGetPoint{X}
    \tkzDefCircle[through](X,P)
    \tkzGetLength{rXPpt}
    \tkzpttocm(\rXPpt){rXPcm}
    \tkzDefPointWith[linear,K=0.3](X,P)
    \tkzGetPoint{x}
    \tkzDefCircle[through](x,P)
    \tkzGetLength{rxPpt}
    \tkzpttocm(\rxPpt){rxPcm}
    \tkzDrawCircles(X,P x,P)
    \tkzDrawPoints(X,x)
    \tkzDrawSegment[<->, >=triangle 45](x,P)
    \tkzDrawSegment(P,X)
    \tkzLabelPoints(X,x)
    \tkzLabelLine[pos=0.5,left](x,P){r}
    \tkzCalcLength[cm](X,P)\tkzGetLength{rXP}
    \tkzDefShiftPoint[X](-90:\rXP){y}
    \tkzDrawSegments[<->, >=triangle 45](X,y)
    \tkzLabelLine[pos=0.5,left](X,y){R}
\end{tikzpicture}
```

19. Definition of circle by transformation; \tkzDefCircleBy

These transformations are:

- translation;
- homothety;
- orthogonal reflection or symmetry;
- central symmetry;
- orthogonal projection;
- rotation (degrees);
- orthogonal from ;
- orthogonal through;
- inversion.

The choice of transformations is made through the options. The macro is `\tkzDefCircleBy` and the other for the transformation of a list of points `\tkzDefCirclesBy`. For example, we'll write:

```
\tkzDefCircleBy[translation= from A to A'](O,M)
```

O is the center and M is a point on the circle. The image is a circle. The new center is `tkzFirstPointResult` and `tkzSecondPointResult` is a point on the new circle. You can get the results with the macro `\tkzGetPoints` .

`\tkzDefCircleBy[⟨local options⟩](⟨pt1,pt2⟩)`

The argument is a couple of points. The results is a couple of points. If you want to keep these points then the macro `\tkzGetPoints{O'}{M'}` allows you to assign the name `O'` to the center and `M'` to the point on the circle.

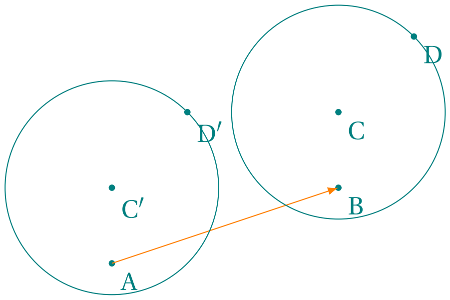
arguments	definition	examples
pt1,pt2	existing points	(O, M)

options		examples
translation	= from #1 to #2	[translation=from A to B](O,M)
homothety	= center #1 ratio #2	[homothety=center A ratio .5](O,M)
reflection	= over #1--#2	[reflection=over A--B](O,M)
symmetry	= center #1	[symmetry=center A](O,M)
projection	= onto #1--#2	[projection=onto A--B](O,M)
rotation	= center #1 angle #2	[rotation=center O angle 30](O,M)
orthogonal from	= #1	[orthogonal from = A](O,M)
orthogonal through	= #1 and #2	[orthogonal through = A and B](O,M)
inversion	= center #1 through #2	[inversion =center O through A](O,M)

The image is only defined and not drawn.

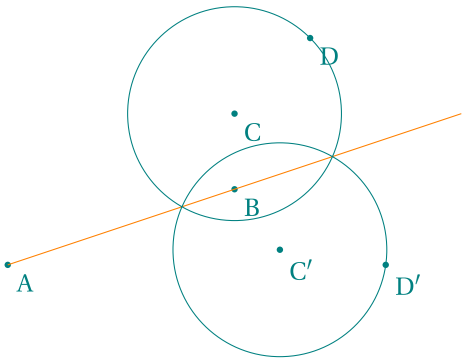
19.1. Examples of transformations

19.1.1. Translation



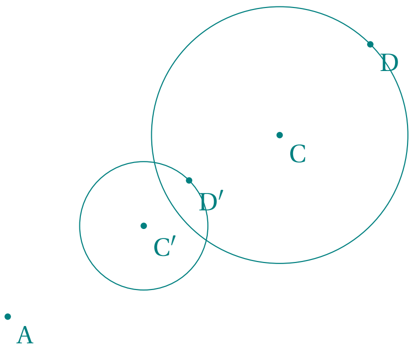

```
\begin{tikzpicture}[>=latex]
 \tkzDefPoint(0,0){A}  \tkzDefPoint(3,1){B}
 \tkzDefPoint(3,2){C}   \tkzDefPoint(4,3){D}
 \tkzDefCircleBy[translation= from B to A](C,D)
 \tkzGetPoints{C'}{D'}
 \tkzDrawPoints[teal](A,B,C,D,C',D')
 \tkzLabelPoints[color=teal](A,B,C,D,C',D')
 \tkzDrawSegments[orange,->](A,B)
 \tkzDrawCircles(C,D C',D')
\end{tikzpicture}
```

19.1.2. Reflection (orthogonal symmetry)



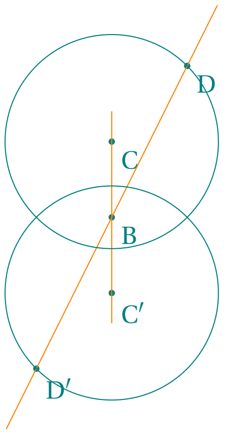

```
\begin{tikzpicture}[>=latex]
 \tkzDefPoint(0,0){A}  \tkzDefPoint(3,1){B}
 \tkzDefPoint(3,2){C}   \tkzDefPoint(4,3){D}
 \tkzDefCircleBy[reflection = over A--B](C,D)
 \tkzGetPoints{C'}{D'}
 \tkzDrawPoints[teal](A,B,C,D,C',D')
 \tkzLabelPoints[color=teal](A,B,C,D,C',D')
 \tkzDrawLine[add =0 and 1][orange](A,B)
 \tkzDrawCircles(C,D C',D')
\end{tikzpicture}
```

19.1.3. Homothety



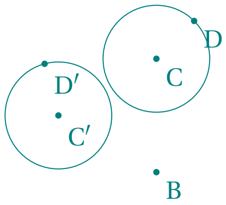

```
\begin{tikzpicture}[scale=1.2]
 \tkzDefPoint(0,0){A}   \tkzDefPoint(3,1){B}
 \tkzDefPoint(3,2){C}   \tkzDefPoint(4,3){D}
 \tkzDefCircleBy[homothety=center A ratio .5](C,D)
 \tkzGetPoints{C'}{D'}
 \tkzDrawPoints[teal](A,C,D,C',D')
 \tkzLabelPoints[color=teal](A,C,D,C',D')
 \tkzDrawCircles(C,D C',D')
\end{tikzpicture}
```

19.1.4. Symmetry




```
\begin{tikzpicture}[scale=1]
 \tkzDefPoint(0,0){A}   \tkzDefPoint(3,1){B}
 \tkzDefPoint(3,2){C}   \tkzDefPoint(4,3){D}
 \tkzDefCircleBy[symmetry=center B](C,D)
 \tkzGetPoints{C'}{D'}
 \tkzDrawPoints[teal](B,C,D,C',D')
 \tkzLabelPoints[color=teal](B,C,D,C',D')
 \tkzDrawLines[orange](C,C' D,D')
 \tkzDrawCircles(C,D C',D')
\end{tikzpicture}
```

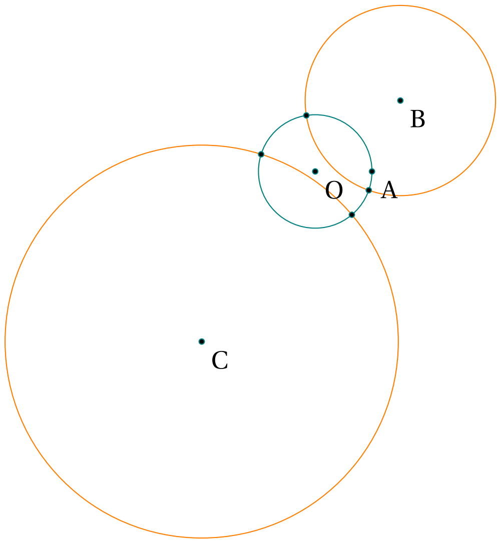
19.1.5. Rotation




```
\begin{tikzpicture}[scale=0.5]
 \tkzDefPoint(0,0){A}   \tkzDefPoint(3,-1){B}
 \tkzDefPoint(3,2){C}   \tkzDefPoint(4,3){D}
 \tkzDefCircleBy[rotation=center B angle 60](C,D)
 \tkzGetPoints{C'}{D'}
 \tkzDrawPoints[teal](B,C,D,C',D')
 \tkzLabelPoints[color=teal](B,C,D,C',D')
 \tkzDrawCircles(C,D C',D')
\end{tikzpicture}
```

19.1.6. Orthogonal from

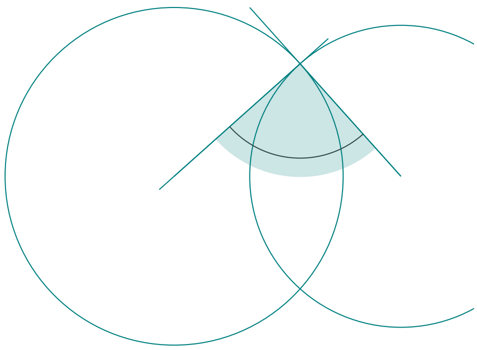
Orthogonal circle of given center. `\tkzGetPointsz1z2` gives two points of the circle.




```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoints{0/0/O,1/0/A}
  \tkzDefPoints{1.5/1.25/B,-2/-3/C}
  \tkzDefCircleBy[orthogonal from=B](O,A)
  \tkzGetPoints{z1}{z2}
  \tkzDefCircleBy[orthogonal from=C](O,A)
  \tkzGetPoints{t1}{t2}
  \tkzDrawCircle(O,A)
  \tkzDrawCircles[new](B,z1 C,t1)
  \tkzDrawPoints(t1,t2,C)
  \tkzDrawPoints(z1,z2,O,A,B)
  \tkzLabelPoints(O,A,B,C)
\end{tikzpicture}
```

19.1.7. Orthogonal from : Right angle between circles

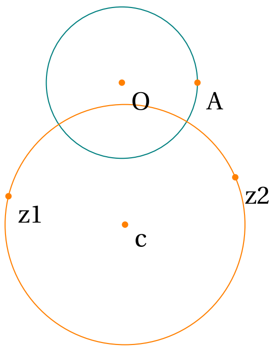
We are looking for a circle orthogonal to the given circle.



```
\begin{tikzpicture}[scale=.5]
\tkzDefPoints{0/0/A,6/0/B,4/2/D}
\tkzDefCircleBy[orthogonal from=B](A,D)
\tkzGetFirstPoint{C}
\tkzDrawCircles(A,C B,C)
\tkzDefTangent[at=C](A) \tkzGetPoint{a}
\tkzDefPointsBy[symmetry = center C](a){d}
\tkzDefTangent[at=C](B) \tkzGetPoint{b}
\tkzDrawLines[add=1 and 4](a,C  C,b)
\tkzDrawSegments(A,C B,C)
\tkzMarkAngle[size=2.5](b,C,d)
\tkzFillAngle[fill=teal,opacity=.2,size=3](b,C,d)
\end{tikzpicture}
```

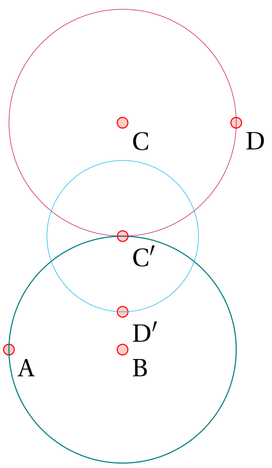
19.1.8. Orthogonal through

Orthogonal circle passing through two given points.




```
\begin{tikzpicture}[scale=1]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(1,0){A}
  \tkzDrawCircle(O,A)
  \tkzDefPoint(-1.5,-1.5){z1}
  \tkzDefPoint(1.5,-1.25){z2}
  \tkzDefCircleBy[orthogonal through=z1 and z2](O,A)
   \tkzGetPoint{c}
  \tkzDrawCircle[new](tkzPointResult,z1)
  \tkzDrawPoints[new](O,A,z1,z2,c)
  \tkzLabelPoints(O,A,z1,z2,c)
\end{tikzpicture}
```

19.1.9. Inversion




```
\begin{tikzpicture}[scale=1.5]
\tkzSetUpPoint[size=4,color=red,fill=red!20]
\tkzSetUpStyle[color=purple,ultra thin]{st1}
\tkzSetUpStyle[color=cyan,ultra thin]{st2}
\tkzDefPoint(2,0){A} \tkzDefPoint(3,0){B}
\tkzDefPoint(3,2){C} \tkzDefPoint(4,2){D}
\tkzDefCircleBy[inversion = center B through A](C,D)
\tkzGetPoints{C'}{D'}
\tkzDrawPoints(A,B,C,D,C',D')
\tkzLabelPoints(A,B,C,D,C',D')
\tkzDrawCircles(B,A)
\tkzDrawCircles[st1](C,D)
\tkzDrawCircles[st2](C',D')
\end{tikzpicture}
```

20. Intersections

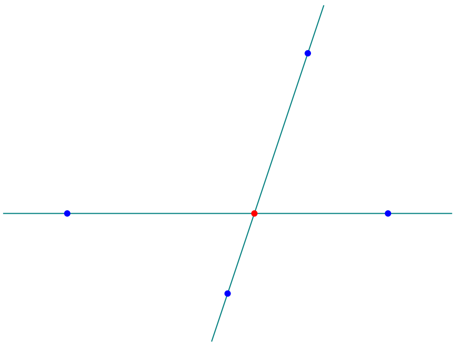
It is possible to determine the coordinates of the points of intersection between two straight lines, a straight line and a circle, and two circles.
The associated commands have no optional arguments and the user must determine the existence of the intersection points himself.

20.1. Intersection of two straight lines

`\tkzInterLL(⟨A, B⟩)(⟨C, D⟩)`

Defines the intersection point **tkzPointResult** of the two lines (AB) and (CD). The known points are given in pairs (two per line) in brackets, and the resulting point can be retrieved with the macro `\tkzDefPoint`.

20.1.1. Example of intersection between two straight lines



```
\begin{tikzpicture}[rotate=-45,scale=.75]
  \tkzDefPoint(2,1){A}
     \tkzDefPoint(6,5){B}
  \tkzDefPoint(3,6){C}
     \tkzDefPoint(5,2){D}
  \tkzDrawLines(A,B C,D)
  \tkzInterLL(A,B)(C,D)
     \tkzGetPoint{I}
  \tkzDrawPoints[color=blue](A,B,C,D)
   \tkzDrawPoint[color=red](I)
\end{tikzpicture}
```

20.2. Intersection of a straight line and a circle

As before, the line is defined by a couple of points. The circle is also defined by a couple:

- (O, C) which is a pair of points, the first is the centre and the second is any point on the circle.
- (O, r) The r measure is the radius measure. The unit can be the *cm* or *pt*.

`\tkzInterLC[⟨options⟩](⟨A, B⟩)(⟨O, C⟩) or (⟨O, r⟩) or (⟨O, C, D⟩)`

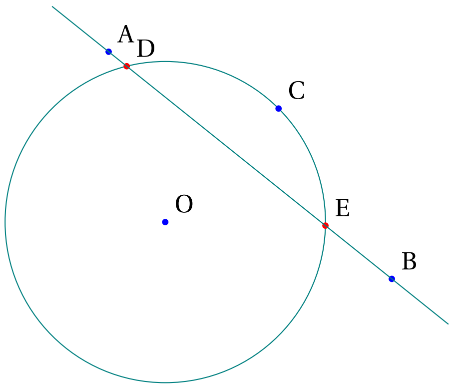
So the arguments are two couples.

options	default	definition
`N`	`N`	`(O,C) determines the circle`
`R`	`N`	`(O, 1 ) unit 1 cm`
`with nodes`	`N`	`(O,C,D) CD is a radius`

The macro defines the intersection points I and J of the line (AB) and the center circle O with radius r if they exist; otherwise, an error will be reported in the `.log` file. **`with nodes`** vous évite de calculer le rayon qui est la longueur de [CD].

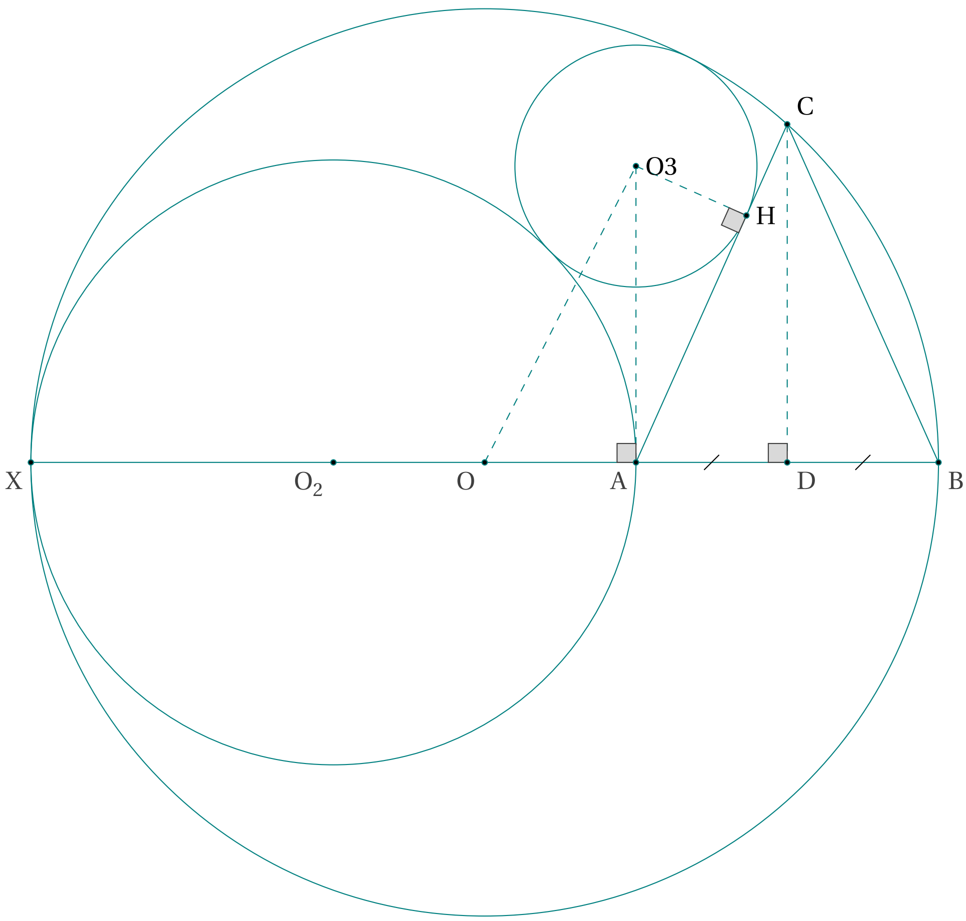
20.2.1. Line-circle intersection

In the following example, the drawing of the circle uses two points and the intersection of the straight line and the circle uses two pairs of points:




```
\begin{tikzpicture}[scale=.75]
 \tkzInit[xmax=5,ymax=4]
 \tkzDefPoint(1,1){O}
 \tkzDefPoint(0,4){A}
 \tkzDefPoint(5,0){B}
 \tkzDefPoint(3,3){C}
 \tkzInterLC(A,B)(O,C)  \tkzGetPoints{D}{E}
 \tkzDrawCircle(O,C)
 \tkzDrawPoints[color=blue](O,A,B,C)
 \tkzDrawPoints[color=red](D,E)
 \tkzDrawLine(A,B)
 \tkzLabelPoints[above right](O,A,B,C,D,E)
\end{tikzpicture}
```

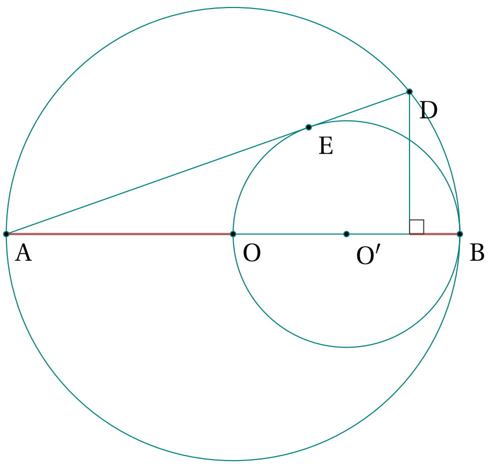
20.2.2. Line-circle intersection in Sangaku



```
\begin{tikzpicture}[scale=1]
 \def\ORadius{6}
 \def\OORadius{4}
 \pgfmathparse{(2*(\ORadius-\OORadius))/(\ORadius/\OORadius+1)}%
 \let\OOORadius\pgfmathresult%
 \pgfmathparse{\ORadius-\OOORadius}%
 \let\OOOORadius\pgfmathresult%
 \pgfmathparse{2*\OORadius-\ORadius}%
 \let\XA\pgfmathresult%
 \tkzDefPoint["$O$" below left](0,0){O}
 \ifdim\XA pt  =  0pt\relax%
     \tkzDefPoint["$A$"  below right](\XA,0){A}
 \else
     \tkzDefPoint["$A$"  below left](\XA,0){A}
 \fi
 \tkzDefPoint["$D$"  below right](\OORadius,0){D}
 \tkzDefPoint["$X$"  below left](-\ORadius,0){X}
 \tkzDefPoint["$B$"  below right](\ORadius,0){B}
 \tkzDefPoint["$O_2$" below left](\OORadius-\ORadius,0){O2}
 \tkzDefLine[mediator](A,B)             \tkzGetPoints{mr}{ml}
 \tkzInterLC[R](D,mr)(O,\ORadius)    \tkzGetPoints{C}{E}
 \tkzDefLine[orthogonal=through A](X,A) \tkzGetPoint{pr}
 \ifdim\XA pt < 0 pt\relax
   \tkzInterLC[R](A,pr)(O,\OOOORadius) \tkzGetPoints{O4}{O3}
 \else
 \ifdim\XA pt = 0pt\relax
   \tkzInterLC[R](A,pr)(O,\OOOORadius) \tkzGetPoints{O4}{O3}
 \else
   \tkzInterLC[R](A,pr)(O,\OOOORadius) \tkzGetPoints{O3}{O4}
 \fi
 \fi
 \tkzDefPointBy[projection=onto A--C](O3) \tkzGetPoint{H}
 \tkzDrawCircles[R](O,{\ORadius} O2,{\OORadius} O3,{\OOORadius})
 \tkzDrawSegments[dashed](O,O3 C,D O3,A O3,H)
 \tkzDrawSegments(X,B A,C B,C)
 \tkzMarkSegments[mark=s|](D,B D,A)
 \tkzLabelPoints[right](O3,H)
 \tkzLabelPoint[above right](C){$C$}
 \tkzMarkRightAngles[fill=gray!30](X,D,C X,A,O3 A,H,O3)
 \tkzDrawPoints(A,B,C,D,X,O,O2,O3,H)
\end{tikzpicture}
```

20.2.3. More complex example of a line-circle intersection

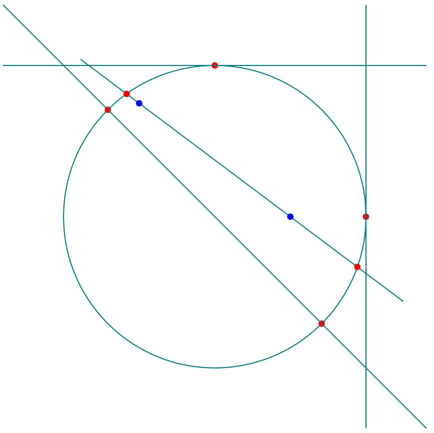
Figure from http://gogeometry.com/problem/p190_tangent_circle




```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(8,0){B}
  \tkzDefMidPoint(A,B)
  \tkzGetPoint{O}
  \tkzDefMidPoint(O,B)
  \tkzGetPoint{O'}
  \tkzDefTangent[from=A](O',B)
  \tkzGetSecondPoint{E}
  \tkzInterLC(A,E)(O,B)
  \tkzGetSecondPoint{D}
  \tkzDefPointBy[projection=onto A--B](D)
  \tkzGetPoint{F}
  \tkzDrawCircles(O,B O',B)
  \tkzDrawSegments(A,D A,B D,F)
  \tkzDrawSegments[color=red,line width=1pt,
      opacity=.4](A,O F,B)
  \tkzDrawPoints(A,B,O,O',E,D)
  \tkzMarkRightAngle(D,F,B)
  \tkzLabelPoints(A,B,O,O',E,D)
\end{tikzpicture}
```

20.2.4. Circle defined by a center and a measure, and special cases

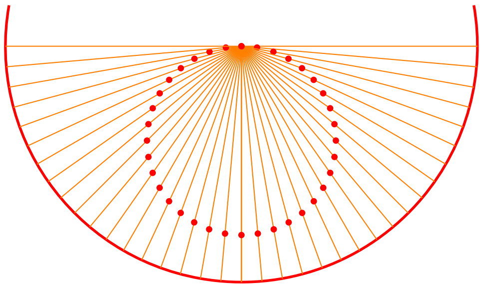
Let's look at some special cases like straight lines tangent to the circle.



```
\begin{tikzpicture}[scale=.5]
  \tkzDefPoint(0,8){A}  \tkzDefPoint(8,0){B}
  \tkzDefPoint(8,8){C}  \tkzDefPoint(4,4){I}
  \tkzDefPoint(2,7){E}  \tkzDefPoint(6,4){F}
  \tkzInterLC[R](A,C)(I,4)  \tkzGetPoints{I1}{I2}
  \tkzInterLC[R](B,C)(I,4)  \tkzGetPoints{J1}{J2}
  \tkzInterLC[R](A,B)(I,4)  \tkzGetPoints{K1}{K2}
  \tkzInterLC[R](E,F)(I,4)  \tkzGetPoints{I2}{J2}
  \tkzDrawCircle[R](I,4)
  \tkzDrawPoints[color=red](I1,J1,K1,K2)
  \tkzDrawLines(A,B B,C A,C I2,J2)
  \tkzDrawPoints[color=blue](E,F)
  \tkzDrawPoints[color=red](I2,J2)
\end{tikzpicture}
```

20.2.5. More complex example

Be careful with the syntax. First of all, calculations for the points can be done during the passage of the arguments, but the syntax of **xfp** must be respected. You can see that I use the term **pi** because `xfp` can work with radians. You can also work with degrees but in this case, you need to use specific commands like `sind` or `cosd`. Furthermore, when calculations require the use of parentheses, they must be inserted in a group... TEX{ ...}.



```
\begin{tikzpicture}[scale=1.25]
\tkzDefPoint(0,1){J}
\tkzDefPoint(0,0){O}
\tkzDrawArc[R,line width=1pt,color=red](J,2.5)(180,0)
\foreach \i in {0,-5,-10,...,-85,-90}{
  \tkzDefPoint({2.5*cosd(\i)},{1+2.5*sind(\i)}){P}
   \tkzDrawSegment[color=orange](J,P)
   \tkzInterLC[R](P,J)(O,1)
   \tkzGetPoints{M}{N}
   \tkzDrawPoints[red](N)
   }
\foreach \i in {-90,-95,...,-175,-180}{
   \tkzDefPoint({2.5*cosd(\i)},{1+2.5*sind(\i)}){P}
   \tkzDrawSegment[color=orange](J,P)
   \tkzInterLC[R](P,J)(O,1)
   \tkzGetPoints{M}{N}
   \tkzDrawPoints[red](M)
   }
\end{tikzpicture}
```

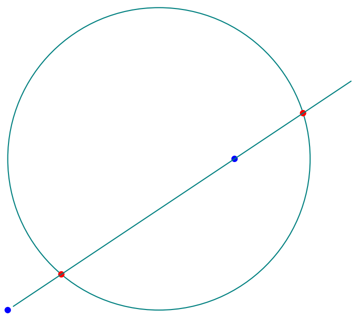
20.2.6. Calculation of radius example 1

With **pgfmath** and **\pgfmathsetmacro**

The radius measurement may be the result of a calculation that is not done within the intersection macro, but before. A length can be calculated in several ways. It is possible of course, to use the module **pgfmath** and the macro **\pgfmathsetmacro**. In some cases, the results obtained are not precise enough, so the following calculation $0.0002 \div 0.0001$ gives 1.98 with pgfmath while xfp will give 2.

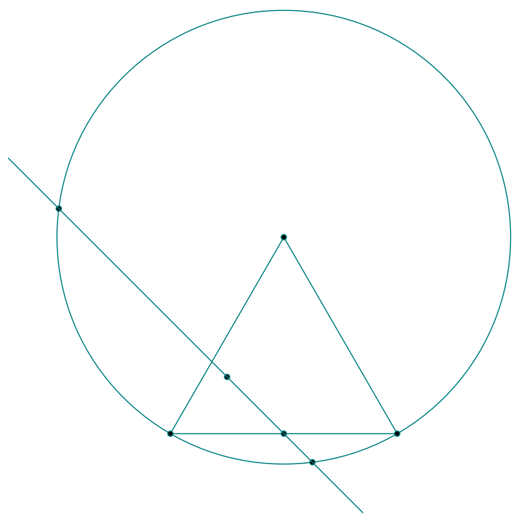
20.2.7. Calculation of radius example 2

With **xfp** and **\fpeval**:



```
 \begin{tikzpicture}
 \tkzDefPoint(2,2){A}
 \tkzDefPoint(5,4){B}
 \tkzDefPoint(4,4){O}
 \pgfmathsetmacro\tkzLen{\fpeval{0.0002/0.0001}}
% or \edef\tkzLen{\fpeval{0.0002/0.0001}}
 \tkzInterLC[R](A,B)(O, \tkzLen)
 \tkzGetPoints{I}{J}
 \tkzDrawCircle[R](O,\tkzLen)
 \tkzDrawPoints[color=blue](A,B)
 \tkzDrawPoints[color=red](I,J)
 \tkzDrawLine(I,J)
\end{tikzpicture}
```

20.2.8. Option "with nodes"



```
\begin{tikzpicture}[scale=.75]
\tkzDefPoints{0/0/A,4/0/B,1/1/D,2/0/E}
\tkzDefTriangle[equilateral](A,B)
\tkzGetPoint{C}
\tkzInterLC[with nodes](D,E)(C,A,B)
\tkzGetPoints{F}{G}
\tkzDrawCircle(C,A)
\tkzDrawPolygon(A,B,C)
\tkzDrawPoints(A,...,G)
\tkzDrawLine(F,G)
\end{tikzpicture}
```

20.3. Intersection of two circles

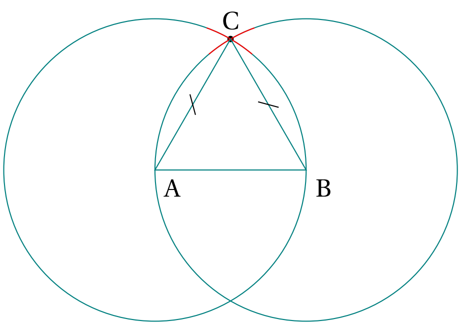
The most frequent case is that of two circles defined by their center and a point, but as before the option `R` allows to use the radius measurements.

`\tkzInterCC[⟨options⟩](⟨O,A⟩)(⟨O',A'⟩) or (⟨O,r⟩)(⟨O',r'⟩) or (⟨O,A,B⟩) (⟨O',C,D⟩)`

options	default	definition
N	N	OA and O'A' are radii, O and O' are the centres
R	N	r and r' are dimensions and measure the radii
with nodes	N	in (A,A,C)(C,B,F) AC and BF give the radii.

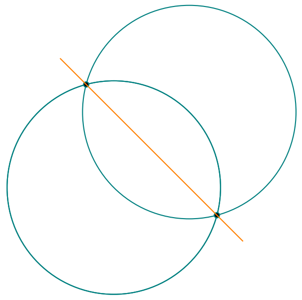
This macro defines the intersection point(s) I and J of the two center circles O and O'. If the two circles do not have a common point then the macro ends with an error that is not handled.
It is also possible to use directly `\tkzInterCCN` and `\tkzInterCCR`.

20.3.1. Construction of an equilateral triangle



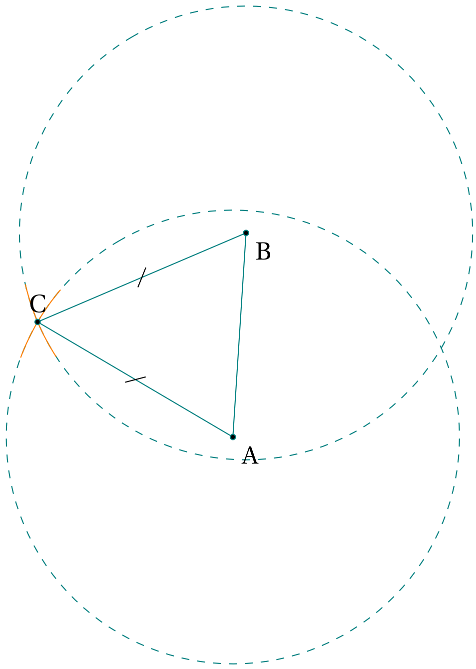

```
\begin{tikzpicture}[trim left=-1cm,scale=.5]
 \tkzDefPoint(1,1){A}
 \tkzDefPoint(5,1){B}
 \tkzInterCC(A,B)(B,A)\tkzGetPoints{C}{D}
 \tkzDrawPoint[color=black](C)
 \tkzDrawCircles(A,B B,A)
 \tkzCompass[color=red](A,C)
 \tkzCompass[color=red](B,C)
 \tkzDrawPolygon(A,B,C)
 \tkzMarkSegments[mark=s|](A,C B,C)
 \tkzLabelPoints[](A,B)
 \tkzLabelPoint[above](C){$C$}
\end{tikzpicture}
```

20.3.2. Example a mediator



```
\begin{tikzpicture}[scale=.5]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(2,2){B}
  \tkzInterCC(B,A)(A,B)\tkzGetPoints{M}{N}
  \tkzDrawCircles[color=teal](A,B B,A)
  \tkzDrawCircle[color=teal](A,B)
  \tkzDrawPoints(M,N)
  \tkzDrawLine[new](M,N)
\end{tikzpicture}
```

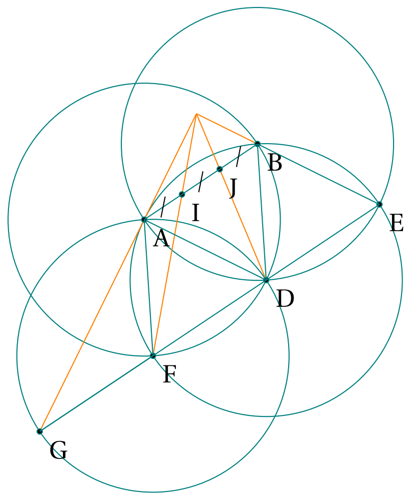
20.3.3. An isosceles triangle.




```
\begin{tikzpicture}[rotate=120,scale=.75]
 \tkzDefPoint(1,2){A}
 \tkzDefPoint(4,0){B}
 \tkzInterCC[R](A,4)(B,4)
 \tkzGetPoints{C}{D}
 \tkzDrawCircles[R,dashed](A,4 B,4)
 \tkzCompass[new](A,C)
 \tkzCompass[new](B,C)
 \tkzDrawPolygon(A,B,C)
 \tkzDrawPoints(A,B,C)
 \tkzMarkSegments[mark=s|](A,C B,C)
 \tkzLabelPoints[](A,B)
 \tkzLabelPoint[above](C){$C$}
\end{tikzpicture}
```

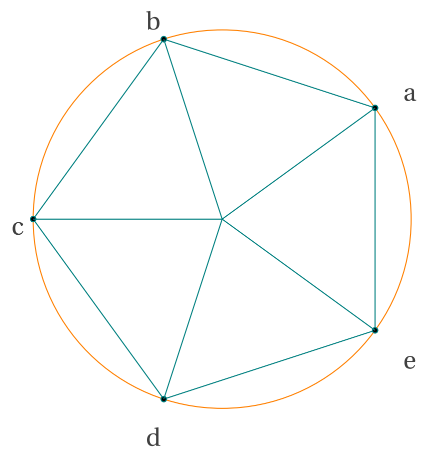
20.3.4. Segment trisection

The idea here is to divide a segment with a ruler and a compass into three segments of equal length.



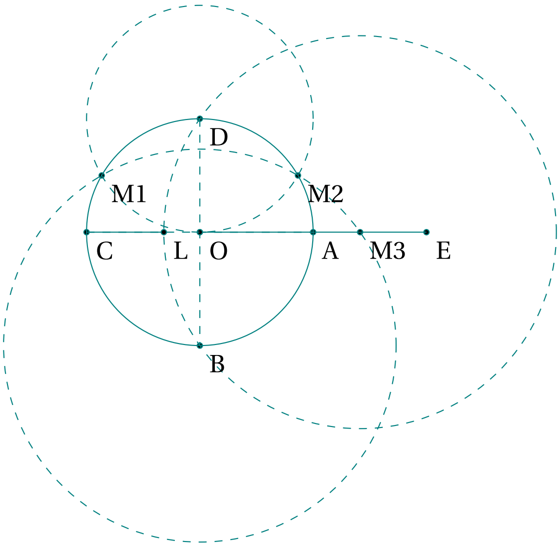

```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoint(0,0){A}
 \tkzDefPoint(3,2){B}
 \tkzInterCC(A,B)(B,A)  \tkzGetPoints{C}{D}
 \tkzInterCC(D,B)(B,A)  \tkzGetPoints{A}{E}
 \tkzInterCC(D,B)(A,B)  \tkzGetPoints{F}{B}
 \tkzInterLC(E,F)(F,A)  \tkzGetPoints{D}{G}
 \tkzInterLL(A,G)(B,E)  \tkzGetPoint{O}
 \tkzInterLL(O,D)(A,B)  \tkzGetPoint{J}
 \tkzInterLL(O,F)(A,B)  \tkzGetPoint{I}
 \tkzDrawCircles(D,A A,B B,A F,A)
 \tkzDrawSegments[new](O,G
  O,B O,D O,F)
 \tkzDrawPoints(A,B,D,E,F,G,I,J)
 \tkzLabelPoints(A,B,D,E,F,G,I,J)
 \tkzDrawSegments(A,B B,D A,D%
  A,F F,G E,G B,E)
 \tkzMarkSegments[mark=s|](A,I I,J J,B)
\end{tikzpicture}
```

20.3.5. With the option "with nodes"



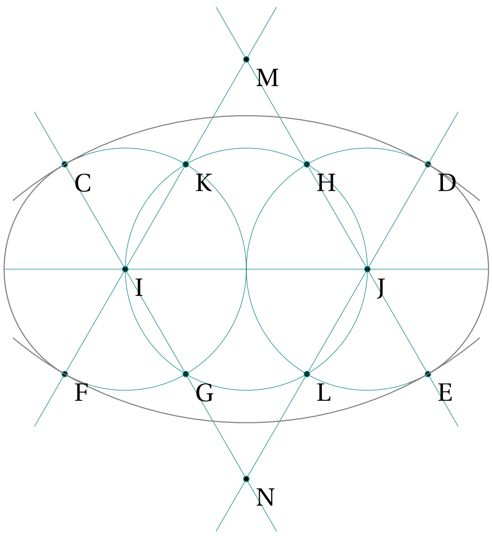

```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoints{0/0/a,0/5/B,5/0/C}
 \tkzDefPoint(54:5){F}
 \tkzInterCC[with nodes](A,A,C)(C,B,F)
 \tkzGetPoints{a}{e}
 \tkzInterCC(A,C)(a,e) \tkzGetFirstPoint{b}
 \tkzInterCC(A,C)(b,a) \tkzGetFirstPoint{c}
 \tkzInterCC(A,C)(c,b) \tkzGetFirstPoint{d}
 \tkzDrawCircle[new](A,C)
 \tkzDrawPoints(a,b,c,d,e)
 \tkzDrawPolygon(a,b,c,d,e)
 \foreach \vertex/\num in {a/36,b/108,c/180,
                          d/252,e/324}{%
 \tkzDrawPoint(\vertex)
 \tkzLabelPoint[label=\num:$\vertex$](\vertex){}
 \tkzDrawSegment(A,\vertex)
 }
\end{tikzpicture}
```

20.3.6. Mix of intersections




```
\begin{tikzpicture}[scale = .75]
  \tkzDefPoint(2,2){A}
  \tkzDefPoint(0,0){B}
  \tkzDefPoint(-2,2){C}
  \tkzDefPoint(0,4){D}
  \tkzDefPoint(4,2){E}
  \tkzCircumCenter(A,B,C)\tkzGetPoint{O}
  \tkzInterCC[R](O,2)(D,2) \tkzGetPoints{M1}{M2}
  \tkzInterCC(O,A)(D,O) \tkzGetPoints{1}{2}
  \tkzInterLC(A,E)(B,M1) \tkzGetSecondPoint{M3}
  \tkzInterLC(O,C)(M3,D) \tkzGetSecondPoint{L}
  \tkzDrawSegments(C,L)
  \tkzDrawPoints(A,B,C,D,E,M1,M2,M3,O,L)
  \tkzDrawSegments(O,E)
  \tkzDrawSegments[dashed](C,A D,B)
  \tkzDrawPoint(O)
  \tkzDrawCircles[dashed](M3,D B,M2 D,O)
  \tkzDrawCircle(O,A)
  \tkzLabelPoints(A,B,C,D,E,M1,M2,M3,O,L)
\end{tikzpicture}
```

20.3.7. An oval




```
\begin{tikzpicture}[scale=0.4]
  \tkzDefPoint(-4,0){I}
  \tkzDefPoint(4,0){J}
  \tkzDefPoint(0,0){O}
  \tkzInterCC(J,O)(O,J) \tkzGetPoints{L}{H}
  \tkzInterCC(I,O)(O,I) \tkzGetPoints{K}{G}
  \tkzInterLL(I,K)(J,H) \tkzGetPoint{M}
  \tkzInterLL(I,G)(J,L) \tkzGetPoint{N}
  \tkzDefPointsBy[symmetry=center J](L,H){D,E}
  \tkzDefPointsBy[symmetry=center I](G,K){C,F}
  \begin{scope}[line style/.style = {very thin,teal}]
    \tkzDrawLines[add=1.5 and 1.5](I,K I,G J,H J,L)
    \tkzDrawLines[add=.5 and .5](I,J)
    \tkzDrawPoints(H,L,K,G,I,J,D,E,C,F,M,N)
    \tkzDrawCircles[R](O,4 I,4 J,4)
    \tkzDrawArc(N,D)(C)
    \tkzDrawArc(M,F)(E)
    \tkzDrawArc(J,E)(D)
    \tkzDrawArc(I,C)(F)
  \end{scope}
  \tkzLabelPoints(H,L,K,G,I,J,D,E,C,F,M,N)
\end{tikzpicture}
```

21. The angles

21.1. Recovering an angle \tkzGetAngle

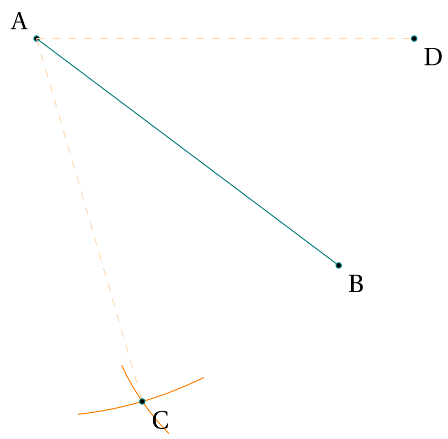
\tkzGetAngle(⟨name of macro⟩)

Assigns the value in degree of an angle to a macro. This macro retrieves \tkzAngleResult and stores the result in a new macro.

arguments	example	explication
name of macro	\tkzGetAngle{ang}	\ang contains the value of the angle.

21.2. Example of the use of \tkzGetAngle

The point here is that (AB) is the bisector of $\widehat{CAD}$, such that the AD slope is zero. We recover the slope of (AB) and then rotate twice.




```
\begin{tikzpicture}
  \tkzDefPoint(1,5){A} \tkzDefPoint(5,2){B}
  \tkzDrawSegment(A,B)
  \tkzFindSlopeAngle(A,B)\tkzGetAngle{tkzang}
  \tkzDefPointBy[rotation= center A angle \tkzang ](B)
   \tkzGetPoint{C}
  \tkzDefPointBy[rotation= center A angle -
\tkzang ](B)
  \tkzGetPoint{D}
  \tkzCompass[length=1](A,C)
  \tkzCompass[delta=10,brown](B,C)
   \tkzDrawPoints(A,B,C,D)
  \tkzLabelPoints(B,C,D)
  \tkzLabelPoints[above left](A)
  \tkzDrawSegments[style=dashed,color=orange!30](A,C A,D)
\end{tikzpicture}
```

21.3. Angle formed by three points

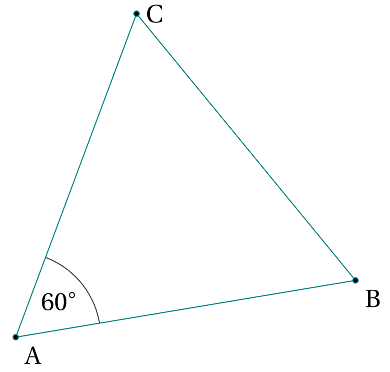
\tkzFindAngle(⟨pt1,pt2,pt3⟩)

The result is stored in a macro \tkzAngleResult.

arguments	example	explication
(pt1,pt2,pt3)	\tkzFindAngle(A,B,C)	\tkzAngleResult gives the angle $(\overrightarrow{BA},\overrightarrow{BC})$

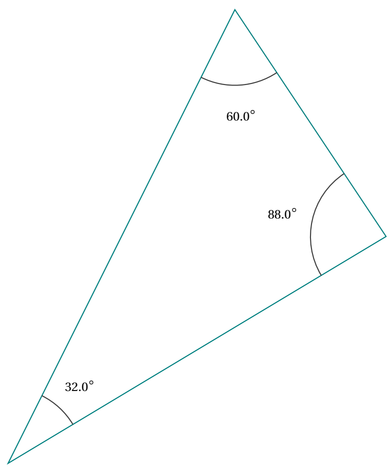
The result is between -180 degrees and +180 degrees. pt2 is the vertex and \tkzGetAngle can retrieve the angle.

21.3.1. Verification of angle measurement



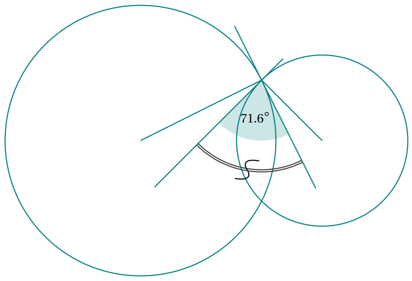

```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoint(-1,1){A}
  \tkzDefPoint(5,2){B}
  \tkzDefEquilateral(A,B)
  \tkzGetPoint{C}
  \tkzDrawPolygon(A,B,C)
  \tkzFindAngle(B,A,C)
  \tkzGetAngle{angleBAC}
  \edef\angleBAC{\fpeval{round(\angleBAC)}}
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(A,B)
  \tkzLabelPoint[right](C){$C$}
  \tkzLabelAngle(B,A,C){\angleBAC$^\circ$}
  \tkzMarkAngle[size=1.5](B,A,C)
\end{tikzpicture}
```

21.3.2. Determination of the three angles of a triangle




```
\begin{tikzpicture}
\tkzDefPoints{0/0/a,5/3/b,3/6/c}
\tkzDrawPolygon(a,b,c)
\tkzFindAngle(c,b,a)
\tkzGetAngle{angleCBA}
\pgfmathparse{round(1+\angleCBA)}
\let\angleCBA\pgfmathresult
\tkzFindAngle(a,c,b)
\tkzGetAngle{angleACB}
\pgfmathparse{round(\angleACB)}
\let\angleACB\pgfmathresult
\tkzFindAngle(b,a,c)
\tkzGetAngle{angleBAC}
\pgfmathparse{round(\angleBAC)}
\let\angleBAC\pgfmathresult
\tkzMarkAngle(c,b,a)
\tkzLabelAngle[pos=1.4](c,b,a)%
              {\tiny $\angleCBA^\circ$}
\tkzMarkAngle(a,c,b)
\tkzLabelAngle[pos=1.4](a,c,b)%
              {\tiny $\angleACB^\circ$}
\tkzMarkAngle(b,a,c)
\tkzLabelAngle[pos=1.4](b,a,c)%
              {\tiny $\angleBAC^\circ$}
\end{tikzpicture}
```

21.3.3. Angle between two circles




```
\begin{tikzpicture}[scale=.4]
\pgfkeys{/pgf/number format/.cd,fixed,precision=1}
\tkzDefPoints{0/0/A,6/0/B,4/2/C}
\tkzDrawCircles(A,C B,C)
\tkzDefTangent[at=C](A) \tkzGetPoint{a}
\tkzDefPointsBy[symmetry = center C](a){d}
\tkzDefTangent[at=C](B) \tkzGetPoint{b}
\tkzDrawLines[add=1 and 4](a,C  C,b)
\tkzDrawSegments(A,C B,C)
\tkzFindAngle(b,C,d)\tkzGetAngle{bcd}
\tkzMarkAngle[size=3,arc=ll,mark=s](b,C,d)
\tkzFillAngle[fill=teal,opacity=.2,size=2](b,C,d)
\tkzLabelAngle[pos=1.25](b,C,d){%
  \tiny $\pgfmathprintnumber{\bcd}^\circ$}
\end{tikzpicture}
```

21.4. Angle formed by a straight line with the horizontal axis \tkzFindSlopeAngle

Much more interesting than the last one. The result is between -180 degrees and +180 degrees.

\tkzFindSlopeAngle(⟨A,B⟩)

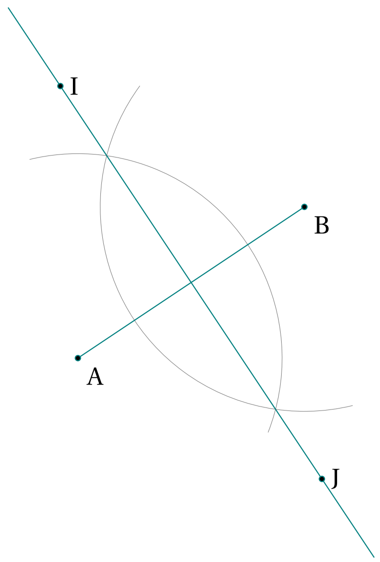
Determines the slope of the straight line (AB). The result is stored in a macro **\tkzAngleResult**.

arguments	example	explication
(pt1,pt2)	**\tkzFindSlopeAngle(A,B)**	

\tkzGetAngle can retrieve the result. If retrieval is not necessary, you can use **\tkzAngleResult**.

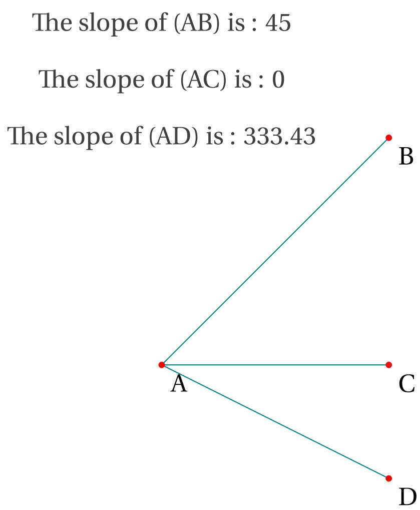
21.4.1. Use of \tkzFindSlopeAngle and \tkzGetAngle

Here is another version of the construction of a mediator




```
\begin{tikzpicture}
 \tkzInit
 \tkzDefPoint(0,0){A}         \tkzDefPoint(3,2){B}
 \tkzDefLine[mediator](A,B)   \tkzGetPoints{I}{J}
 \tkzCalcLength(A,B)          \tkzGetLength{dAB}
 \tkzFindSlopeAngle(A,B)      \tkzGetAngle{tkzangle}
 \begin{scope}[rotate=\tkzangle]
   \tkzSetUpArc[color=gray,line width=0.2pt,/tkzcompass/delta=10]
   \tkzDrawArc[R,arc](B,3/4*\dAB)(120,240)
   \tkzDrawArc[R,arc](A,3/4*\dAB)(-45,60)
   \tkzDrawLine(I,J)          \tkzDrawSegment(A,B)
  \end{scope}
  \tkzDrawPoints(A,B,I,J)     \tkzLabelPoints(A,B)
   \tkzLabelPoints[right](I,J)
\end{tikzpicture}
```

21.4.2. Another use of `\tkzFindSlopeAngle`




```
\begin{tikzpicture}[scale=1.5]
  \tkzDefPoint(1,2){A}    \tkzDefPoint(3,4){B}
  \tkzDefPoint(3,2){C}    \tkzDefPoint(3,1){D}
  \tkzDrawSegments(A,B A,C A,D)
  \tkzDrawPoints[color=red](A,B,C,D)
  \tkzLabelPoints(A,B,C,D)
  \tkzFindSlopeAngle(A,B)\tkzGetAngle{SAB}
  \tkzFindSlopeAngle(A,C)\tkzGetAngle{SAC}
  \tkzFindSlopeAngle(A,D)\tkzGetAngle{SAD}
  \pgfkeys{/pgf/number format/.cd,fixed,precision=2}
  \tkzText(1,5){The slope of (AB) is :
     $\pgfmathprintnumber{\SAB}$}
  \tkzText(1,4.5){The slope of (AC) is :
     $\pgfmathprintnumber{\SAC}$}
  \tkzText(1,4){The slope of (AD) is :
     $\pgfmathprintnumber{\SAD}$}
\end{tikzpicture}
```

22. Random point definition

At the moment there are four possibilities:

1. point in a rectangle;
2. on a segment;
3. on a straight line;
4. on a circle.

22.1. Obtaining random points

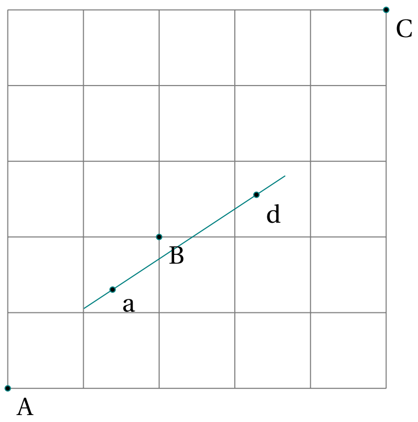
This is the new version that replaces `\tkzGetRandPointOn`.

`\tkzDefRandPointOn[⟨local options⟩]`

The result is a point with a random position that can be named with the macro `\tkzGetPoint`. It is possible to use `tkzPointResult` if it is not necessary to retain the results.

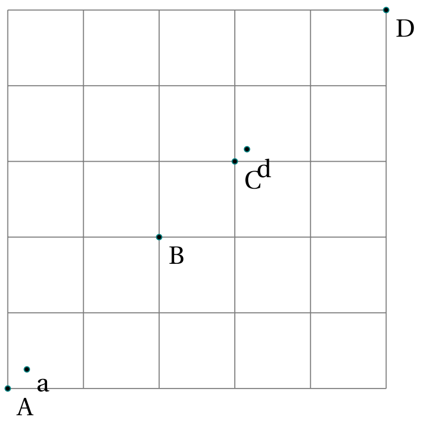
options	default	definition
rectangle=pt1 and pt2		[rectangle=A and B]
segment= pt1--pt2		[segment=A--B]
line=pt1--pt2		[line=A--B]
circle =center pt1 radius dim		[circle = center A radius 2]
circle through=center pt1 through pt2		[circle through= center A through B]
disk through=center pt1 through pt2		[disk through=center A through B]

22.2. Random point in a rectangle



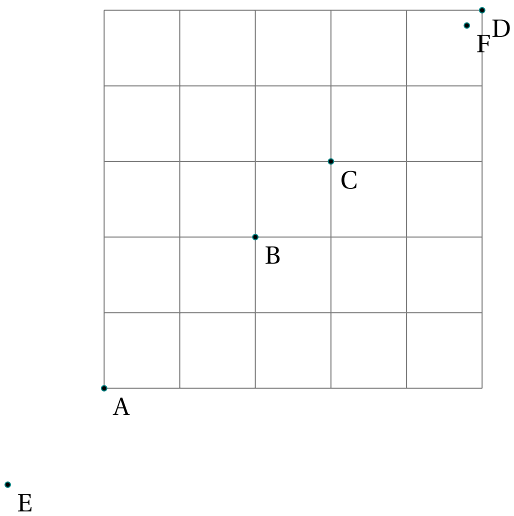

```
\begin{tikzpicture}
  \tkzInit[xmax=5,ymax=5]\tkzGrid
  \tkzDefPoints{0/0/A,2/2/B,5/5/C}
  \tkzDefRandPointOn[rectangle = A and B]
  \tkzGetPoint{a}
  \tkzDefRandPointOn[rectangle = B and C]
  \tkzGetPoint{d}
  \tkzDrawLine(a,d)
  \tkzDrawPoints(A,B,C,a,d)
  \tkzLabelPoints(A,B,C,a,d)
\end{tikzpicture}
```

22.3. Random point on a segment



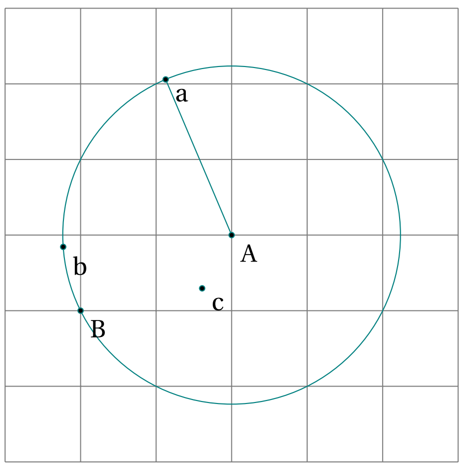

```
\begin{tikzpicture}
  \tkzInit[xmax=5,ymax=5] \tkzGrid
  \tkzDefPoints{0/0/A,2/2/B,3/3/C,5/5/D}
  \tkzDefRandPointOn[segment = A--B]\tkzGetPoint{a}
  \tkzDefRandPointOn[segment = C--D]\tkzGetPoint{d}
  \tkzDrawPoints(A,B,C,D,a,d)
  \tkzLabelPoints(A,B,C,D,a,d)
\end{tikzpicture}
```

22.4. Random point on a straight line



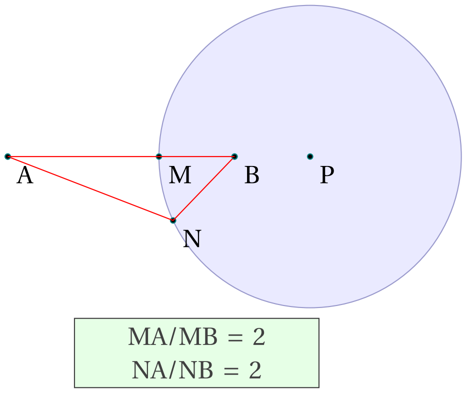

```
\begin{tikzpicture}
  \tkzInit[xmax=5,ymax=5] \tkzGrid
  \tkzDefPoints{0/0/A,2/2/B,3/3/C,5/5/D}
  \tkzDefRandPointOn[line = A--B]\tkzGetPoint{E}
  \tkzDefRandPointOn[line = C--D]\tkzGetPoint{F}
  \tkzDrawPoints(A,...,F)
  \tkzLabelPoints(A,...,F)
\end{tikzpicture}
```

22.4.1. Random point on a circle




```
\begin{tikzpicture}
\tkzInit[ymin=-1,xmax=6,ymax=5] \tkzGrid
\tkzDefPoints{3/2/A,1/1/B}
\tkzCalcLength(A,B) \tkzGetLength{rAB}
\tkzDefRandPointOn[circle = center A radius \rAB]
\tkzGetPoint{a}
\tkzDefRandPointOn[circle through= center A through B]
\tkzGetPoint{b}
\tkzDefRandPointOn[disk through=center A through B]
\tkzGetPoint{c}
\tkzDrawCircle[R](A,\rAB)
\tkzDrawSegment(A,a)
\tkzDrawPoints(A,B,a,b,c)
\tkzLabelPoints(A,B,a,b,c)
\end{tikzpicture}
```

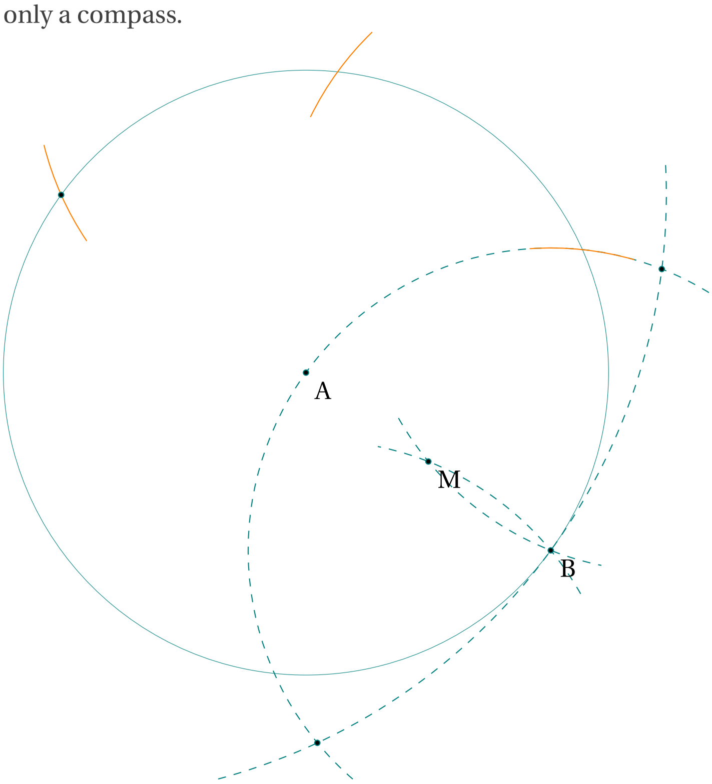
22.4.2. Random example and circle of Apollonius




```
\begin{tikzpicture}[scale=1]
 \tkzDefPoints{0/0/A,3/0/B}
 \def\coeffK{2}
 \tkzApolloniusCenter[K=\coeffK](A,B)
 \tkzGetPoint{P}
 \tkzDefApolloniusPoint[K=\coeffK](A,B)
 \tkzGetPoint{M}
 \tkzDefRandPointOn[circle through=%
               center P through M]
 \tkzGetPoint{N}
 \tkzDefApolloniusRadius[K=\coeffK](A,B)
 \tkzDrawCircle[R,color = blue!50!black,
     fill=blue!20,
     opacity=.4](tkzPointResult,\tkzLengthResult)
 \tkzLabelCircle[R,draw,fill=green!10,%
     text width=3cm,%
     text centered](P,\tkzLengthResult+1)(-120)%
  { $MA/MB=\coeffK$\\$NA/NB=\coeffK$}
 \tkzDrawPoints(A,B,P,M,N)
 \tkzLabelPoints(A,B,P,M,N)
 \tkzDrawSegments[red](N,A N,B)
 \tkzDrawPoints(A,B)
 \tkzDrawSegments[red](A,B)
\end{tikzpicture}
```

22.5. Middle of a compass segment

To conclude this section, here is a more complex example. It involves determining the middle of a segment, using only a compass.



```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDefRandPointOn[circle= center A radius 4]
  \tkzGetPoint{B}
  \tkzDefPointBy[rotation= center A angle 180](B)
  \tkzGetPoint{C}
  \tkzInterCC[R](A,4)(B,4)
  \tkzGetPoints{I}{I'}
  \tkzInterCC[R](A,4)(I,4)
  \tkzGetPoints{J}{B}
  \tkzInterCC(B,A)(C,B)
  \tkzGetPoints{D}{E}
  \tkzInterCC(D,B)(E,B)
  \tkzGetPoints{M}{M'}
  \tkzSetUpArc[ccolor=teal,style=dashed,delta=10]
  \tkzDrawArc(C,D)(E)
  \tkzDrawArc(B,E)(D)
  \tkzDrawCircle[color=teal,line width=.2pt](A,B)
  \tkzDrawArc(D,B)(M)
  \tkzDrawArc(E,M)(B)
  \tkzCompasss[color=orange,style=solid](B,I I,J J,C)
  \tkzDrawPoints(A,B,C,D,E,M)
  \tkzLabelPoints(A,B,M)
 \end{tikzpicture}
```

Part IV.

Drawing and Filling

23. Drawing

tkz-euclide can draw 5 types of objects : point, line or line segment, circle, arc and sector.

23.1. Draw a point or some points

There are two possibilities : `\tkzDrawPoint` for a single point or `\tkzDrawPoints` for one or more points.

23.1.1. Drawing points \tkzDrawPoint

`\tkzDrawPoint[⟨local options⟩](⟨name⟩)`

arguments	default	definition
name of point	no default	Only one point name is accepted

The argument is required. The disc takes the color of the circle, but lighter. It is possible to change everything. The point is a node and therefore it is invariant if the drawing is modified by scaling.

options	default	definition
TikZ options		all TikZ options are valid.
shape	circle	Possible **cross** or **cross out**
size	6	6× \pgflinewidth
color	black	the default color can be changed

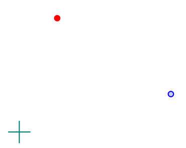
We can create other forms such as **cross**

By default, **point style** is defined like this :

```
\tikzset{point style/.style = {%
          draw          = black,
          inner sep     = 0pt,
          shape         = circle,
          minimum size = 3 pt,
          fill          = black
                              }
       }
```

23.1.2. Example of point drawings

Note that **scale** does not affect the shape of the dots. Which is normal. Most of the time, we are satisfied with a single point shape that we can define from the beginning, either with a macro or by modifying a configuration file.



```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoint(1,3){A}
 \tkzDefPoint(4,1){B}
 \tkzDefPoint(0,0){O}
 \tkzDrawPoint[color=red](A)
 \tkzDrawPoint[fill=blue!20,draw=blue](B)
 \tkzDrawPoint[shape=cross,size=8pt,color=teal](O)
\end{tikzpicture}
```

It is possible to draw several points at once but this macro is a little slower than the previous one. Moreover, we have to make do with the same options for all the points.

\tkzDrawPoints[⟨local options⟩](⟨liste⟩)

arguments	default	definition
points list	no default	example \tkzDrawPoints(A,B,C)

options	default	definition
shape	circle	Possible **cross** or **cross out**
size	6	6× \pgflinewidth
color	black	the default color can be changed

Beware of the final "s", an oversight leads to cascading errors if you try to draw multiple points. The options are the same as for the previous macro.

23.1.3. Example

```
\begin{tikzpicture}
\tkzDefPoints{1/3/A,4/1/B,0/0/C}
\tkzDrawPoints[size=3,color=red,fill=red!50](A,B,C)
\end{tikzpicture}
```

24. Drawing the lines

The following macros are simply used to draw, name lines.

24.1. Draw a straight line

To draw a normal straight line, just give a couple of points. You can use the **add** option to extend the line (This option is due to **Mark Wibrow**, see the code below).
The style of a line is by default :

```
\tikzset{line style/.style = {%
  line width = 0.6pt,
  color      = black,
  style      = solid,
  add        = {.2} and  {.2}%
 }}
```

with

```
\tikzset{%
  add/.style args={#1 and #2}{
      to path={%
($(\tikztostart)!-#1!(\tikztotarget)$)--($(\tikztotarget)!-#2!(\tikztostart)$)%
\tikztonodes}}}
```

You can modify this style with **\tkzSetUpLine** see 41.0.1

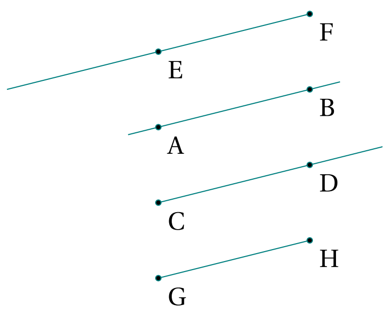
`\tkzDrawLine[⟨local options⟩](⟨pt1,pt2⟩)`

The arguments are a list of two points or three points. It would be possible, as for a half line, to create a style with `\add`.

options	default	definition
`TikZ options`		`all TikZ options are valid.`
`add`	`0.2 and 0.2`	`add = kl and kr, …`
`…`	`…`	`allows the segment to be extended to the left and right.`

`add` defines the length of the line passing through the points pt1 and pt2. Both numbers are percentages. The styles of Ti*k*Z are accessible for plots.

24.1.1. Examples with `add`



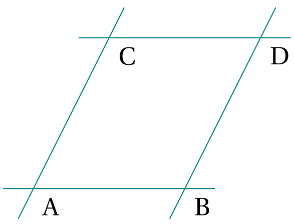

```
\begin{tikzpicture}
 \tkzInit[xmin=-2,xmax=3,ymin=-2.25,ymax=2.25]
 \tkzClip[space=.25]
 \tkzDefPoint(0,0){A} \tkzDefPoint(2,0.5){B}
 \tkzDefPoint(0,-1){C}\tkzDefPoint(2,-0.5){D}
 \tkzDefPoint(0,1){E} \tkzDefPoint(2,1.5){F}
 \tkzDefPoint(0,-2){G} \tkzDefPoint(2,-1.5){H}
 \tkzDrawLine(A,B)    \tkzDrawLine[add = 0 and .5](C,D)
 \tkzDrawLine[add = 1 and 0](E,F)
 \tkzDrawLine[add = 0 and 0](G,H)
 \tkzDrawPoints(A,B,C,D,E,F,G,H)
 \tkzLabelPoints(A,B,C,D,E,F,G,H)
\end{tikzpicture}
```

It is possible to draw several lines, but with the same options.

`\tkzDrawLines[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩)`

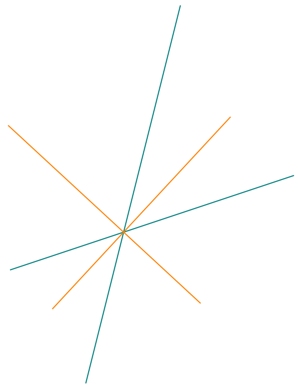
Arguments are a list of pairs of points separated by spaces. The styles of Ti*k*Z are available for the draws.

24.1.2. Example with `\tkzDrawLines`




```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(2,0){B}
  \tkzDefPoint(1,2){C}
  \tkzDefPoint(3,2){D}
  \tkzDrawLines(A,B C,D A,C B,D)
  \tkzLabelPoints(A,B,C,D)
\end{tikzpicture}
```

24.1.3. Example with the option add



```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoint(0,0){O}
 \tkzDefPoint(3,1){I}
 \tkzDefPoint(1,4){J}
 \tkzDefLine[bisector](I,O,J)
   \tkzGetPoint{i}
 \tkzDefLine[bisector out](I,O,J)
   \tkzGetPoint{j}
 \tkzDrawLines[add = 1 and .5](O,I O,J)
 \tkzDrawLines[add = 1 and .5,new](O,i O,j)
\end{tikzpicture}
```

25. Drawing a segment

There is, of course, a macro to simply draw a segment.

25.1. Draw a segment \tkzDrawSegment

`\tkzDrawSegment[⟨local options⟩](⟨pt1,pt2⟩)`

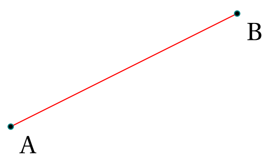
The arguments are a list of two points. The styles of Ti*k*Z are available for the drawings.

argument	example	definition
(pt1,pt2)	(A,B)	draw the segment [A, B]

options	example	definition
Ti*k*Z options		all Ti*k*Z options are valid.
dim	no default	dim = {label,dim,option}, …
…	…	allows you to add dimensions to a figure.

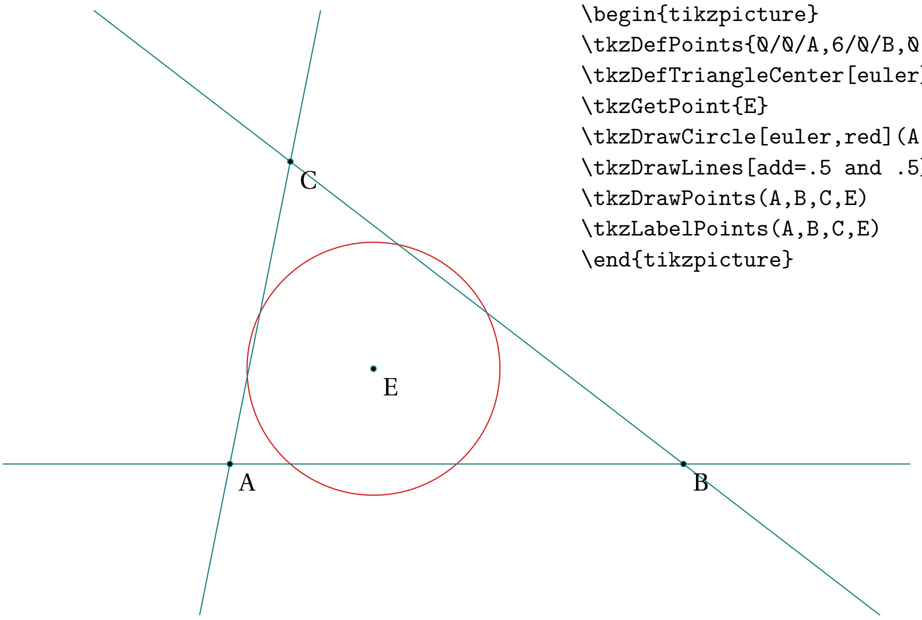
This is of course equivalent to `\draw (A)--(B);`. You can also use the option **add**.

25.1.1. Example with point references



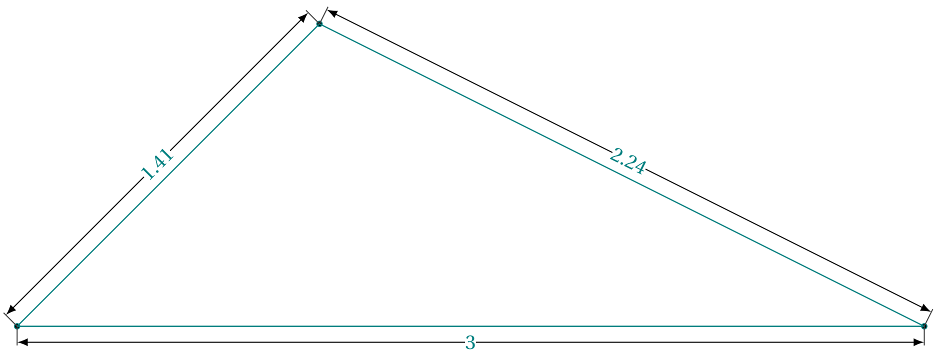

```
\begin{tikzpicture}[scale=1.5]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(2,1){B}
  \tkzDrawSegment[color=red,thin](A,B)
  \tkzDrawPoints(A,B)
  \tkzLabelPoints(A,B)
\end{tikzpicture}
```

25.1.2. Example of extending an segment with option add



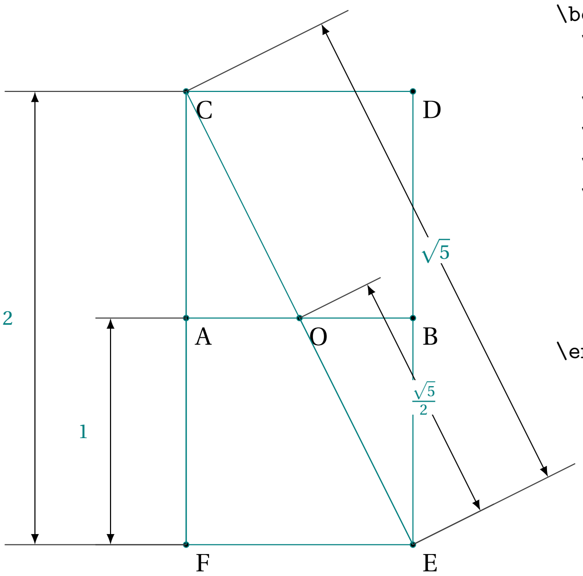

```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,6/0/B,0.8/4/C}
\tkzDefTriangleCenter[euler](A,B,C)
\tkzGetPoint{E}
\tkzDrawCircle[euler,red](A,B,C)
\tkzDrawLines[add=.5 and .5](A,B A,C B,C)
\tkzDrawPoints(A,B,C,E)
\tkzLabelPoints(A,B,C,E)
\end{tikzpicture}
```

25.1.3. Adding dimensions with option dim partI




```
\begin{tikzpicture}[scale=4]
 \pgfkeys{/pgf/number format/.cd,fixed,precision=2}
 % Define the first two points
 \tkzDefPoint(0,0){A}
 \tkzDefPoint(3,0){B}
 \tkzDefPoint(1,1){C}
 % Draw the triangle and the points
 \tkzDrawPolygon(A,B,C)
 \tkzDrawPoints(A,B,C)
 % Label the sides
 \tkzCalcLength(A,B)\tkzGetLength{ABl}
 \tkzCalcLength(B,C)\tkzGetLength{BCl}
 \tkzCalcLength(A,C)\tkzGetLength{ACl}
 % add dim
 \tkzDrawSegment[dim={\pgfmathprintnumber\BCl,6pt,transform shape}](C,B)
 \tkzDrawSegment[dim={\pgfmathprintnumber\ACl,6pt,transform shape}](A,C)
 \tkzDrawSegment[dim={\pgfmathprintnumber\ABl,-6pt,transform shape}](A,B)
\end{tikzpicture}
```

25.1.4. Adding dimensions with option dim part II

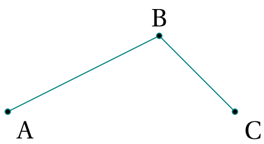



```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoints{0/0/O,-2/0/A,2/0/B,
              -2/4/C,2/4/D,2/-4/E,-2/-4/F}
  \tkzDrawPolygon(C,...,F)
  \tkzDrawSegments(A,B)
  \tkzDrawPoints(A,...,F,O)
  \tkzLabelPoints(A,...,F,O)
   \tkzDrawSegment[dim={ $\sqrt{5}$,2cm,}](C,E)
   \tkzDrawSegment[dim={ $\frac{\sqrt{5}}{2}$,1cm,}](O,E)
   \tkzDrawSegment[dim={ $2$,2cm,left=8pt}](F,C)
   \tkzDrawSegment[dim={ $1$,1cm,left=8pt}](F,A)
\end{tikzpicture}
```

25.2. Drawing segments \tkzDrawSegments

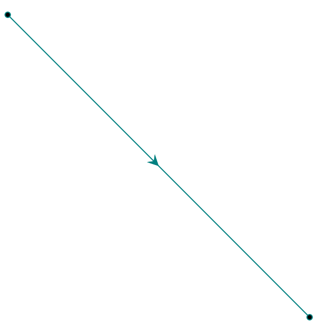
If the options are the same we can plot several segments with the same macro.

\tkzDrawSegments[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩)
The arguments are a two-point couple list. The styles of TikZ are available for the plots.




```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=3,ymin=-1,ymax=2]
  \tkzClip[space=1]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(2,1){B}
  \tkzDefPoint(3,0){C}
  \tkzDrawSegments(A,B B,C)
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(A,C)
  \tkzLabelPoints[above](B)
\end{tikzpicture}
```

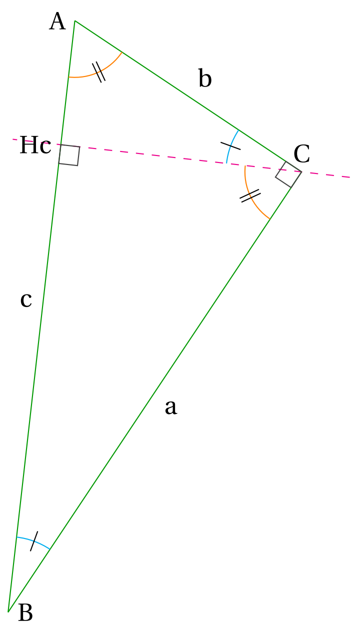
25.2.1. Place an arrow on segment



```
\begin{tikzpicture}
\tkzSetUpStyle[postaction=decorate,
    decoration={markings,
    mark=at position .5 with {\arrow[thick]{#1}}
      }]{myarrow}
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(4,-4){B}
  \tkzDrawSegments[myarrow=stealth](A,B)
  \tkzDrawPoints(A,B)
\end{tikzpicture}
```

25.3. Drawing line segment of a triangle

25.3.1. How to draw Altitude




```
\begin{tikzpicture}[rotate=-90]
\tkzDefPoint(0,1){A}
\tkzDefPoint(2,4){C}
\tkzDefPointWith[orthogonal normed,K=7](C,A)
\tkzGetPoint{B}
\tkzDefSpcTriangle[orthic,name=H](A,B,C){a,b,c}
\tkzDrawLine[dashed,color=magenta](C,Hc)
\tkzDrawSegment[green!60!black](A,C)
\tkzDrawSegment[green!60!black](C,B)
\tkzDrawSegment[green!60!black](B,A)
\tkzLabelPoint[left](A){$A$}
\tkzLabelPoint[right](B){$B$}
\tkzLabelPoint[above](C){$C$}
\tkzLabelPoint[left](Hc){$Hc$}
\tkzLabelSegment[auto](B,A){$c$}
\tkzLabelSegment[auto,swap](B,C){$a$}
\tkzLabelSegment[auto,swap](C,A){$b$}
\tkzMarkAngle[size=1,color=cyan,mark=|](C,B,A)
\tkzMarkAngle[size=1,color=cyan,mark=|](A,C,Hc)
\tkzMarkAngle[size=0.75,
              color=orange,mark=||](Hc,C,B)
\tkzMarkAngle[size=0.75,
              color=orange,mark=||](B,A,C)
\tkzMarkRightAngle(A,C,B)
\tkzMarkRightAngle(B,Hc,C)
\end{tikzpicture}
```

25.4. Drawing a polygon

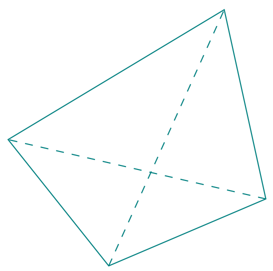
`\tkzDrawPolygon[⟨local options⟩](⟨points list⟩)`

Just give a list of points and the macro plots the polygon using the Ti*k*Z options present. You can replace (A, B, C, D, E) by (A, ..., E) and (P_1, P_2, P_3, P_4, P_5) by (P_1, P..., P_5)

arguments	example	explication
`(⟨pt1,pt2,pt3,...⟩)`	`\tkzDrawPolygon[gray,dashed](A,B,C)`	Drawing a triangle

options	default	example
Options TikZ	...	`\tkzDrawPolygon[red,line width=2pt](A,B,C)`

25.4.1. \tkzDrawPolygon



```
\begin{tikzpicture} [rotate=18,scale=1]
 \tkzDefPoints{0/0/A,2.25/0.2/B,2.5/2.75/C,-0.75/2/D}
 \tkzDrawPolygon(A,B,C,D)
 \tkzDrawSegments[style=dashed](A,C B,D)
\end{tikzpicture}
```

25.4.2. Option two angles

```
\begin{tikzpicture}
\tkzDefPoint(0,0){A}
\tkzDefPoint(6,0){B}
\tkzDefTriangle[two angles = 50 and 70](A,B) \tkzGetPoint{C}
\tkzLabelAngle[pos=1.4](B,A,C){$50^\circ$}
\tkzLabelAngle[pos=0.8](C,B,A){$70^\circ$}
\end{tikzpicture}
```

50° 70°

25.4.3. Style of line

```
\begin{tikzpicture}[scale=.6]
\tkzSetUpLine[line width=5mm,color=teal]
\tkzDefPoint(0,0){O}
\foreach \i in {0,...,5}{%
 \tkzDefPoint({30+60*\i}:4){p\i}}
\tkzDefMidPoint(p1,p3) \tkzGetPoint{m1}
\tkzDefMidPoint(p3,p5) \tkzGetPoint{m3}
\tkzDefMidPoint(p5,p1) \tkzGetPoint{m5}
\tkzDrawPolygon[line join=round](p1,p3,p5)
\tkzDrawPolygon[teal!80,
line join=round](p0,p2,p4)
\tkzDrawSegments(m1,p3 m3,p5 m5,p1)
\tkzDrawCircle[teal,R](O,4.8)
\end{tikzpicture}
```

25.5. Drawing a polygonal chain

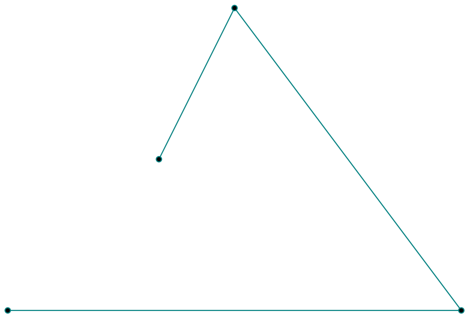
`\tkzDrawPolySeg[⟨local options⟩](⟨points list⟩)`

Just give a list of points and the macro plots the polygonal chain using the Ti*k*Z options present.

arguments	example	explication
`(⟨pt1,pt2,pt3,...⟩)`	`\tkzDrawPolySeg[gray,dashed](A,B,C)`	Drawing a triangle

options	default	example
Options TikZ	...	`\tkzDrawPolySeg[red,line width=2pt](A,B,C)`

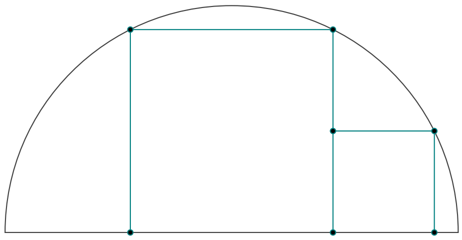
25.5.1. Polygonal chain



```
\begin{tikzpicture}
 \tkzDefPoints{0/0/A,6/0/B,3/4/C,2/2/D}
 \tkzDrawPolySeg(A,...,D)
 \tkzDrawPoints(A,...,D)
\end{tikzpicture}
```

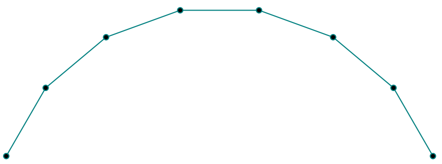
25.5.2. The idea is to inscribe two squares in a semi-circle.

A Sangaku look! It is a question of proving that one can inscribe in a half-disc, two squares, and to determine the length of their respective sides according to the radius.



```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoints{0/0/A,8/0/B,4/0/I}
  \tkzDefSquare(A,B)     \tkzGetPoints{C}{D}
  \tkzInterLC(I,C)(I,B) \tkzGetPoints{E'}{E}
  \tkzInterLC(I,D)(I,B) \tkzGetPoints{F'}{F}
  \tkzDefPointsBy[projection=onto A--B](E,F){H,G}
  \tkzDefPointsBy[symmetry = center H](I){J}
  \tkzDefSquare(H,J)     \tkzGetPoints{K}{L}
  \tkzDrawSector(I,B)(A)
  \tkzDrawPolySeg(H,E,F,G)
  \tkzDrawPolySeg(J,K,L)
  \tkzDrawPoints(E,G,H,F,J,K,L)
\end{tikzpicture}
```

25.5.3. Polygonal chain: index notation



```
\begin{tikzpicture}
\foreach \pt in {1,2,...,8} {%
\tkzDefPoint(\pt*20:3){P_\pt}}
\tkzDrawPolySeg(P_1,P_...,P_8)
\tkzDrawPoints(P_1,P_...,P_8)
\end{tikzpicture}
```

26. Draw a circle with \tkzDrawCircle

26.1. Draw one circle

`\tkzDrawCircle[⟨local options⟩](⟨A,B⟩)`

Attention you need only two points to define a radius or a diameter. An additional option **R** is available to give a measure directly.

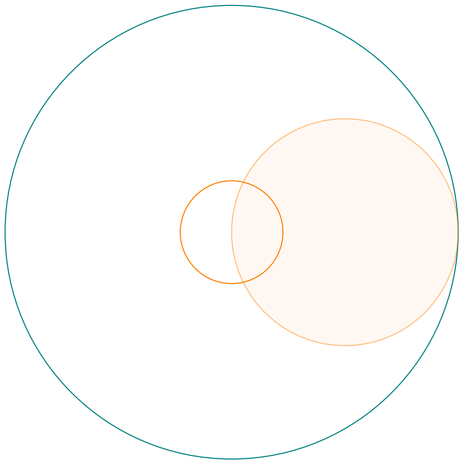
arguments	example	explication
(⟨pt1,pt2⟩)	(⟨A,B⟩)	two points to define a radius or a diameter

options	default	definition
through	through	circle with two points defining a radius
diameter	through	circle with two points defining a diameter
R	through	circle characterized by a point and the measurement of a radius

Of course, you have to add all the styles of TikZ for the tracings...

26.1.1. Circles and styles, draw a circle and color the disc

We'll see that it's possible to colour in a disc while tracing the circle.



```
\begin{tikzpicture}
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(3,0){A}
 % circle with centre O and passing through A
  \tkzDrawCircle(O,A)
 % diameter circle $[OA]$
  \tkzDrawCircle[diameter,new,%
                 line width=.4pt,fill=orange!10,%
                 opacity=.5](O,A)
 % circle with centre O and radius = exp(1) cm
  \edef\rayon{\fpeval{0.25*exp(1)}}
  \tkzDrawCircle[R,color=orange](O,\rayon)
\end{tikzpicture}
```

26.2. Drawing circles

`\tkzDrawCircles[⟨local options⟩](⟨A,B C,D …⟩)`

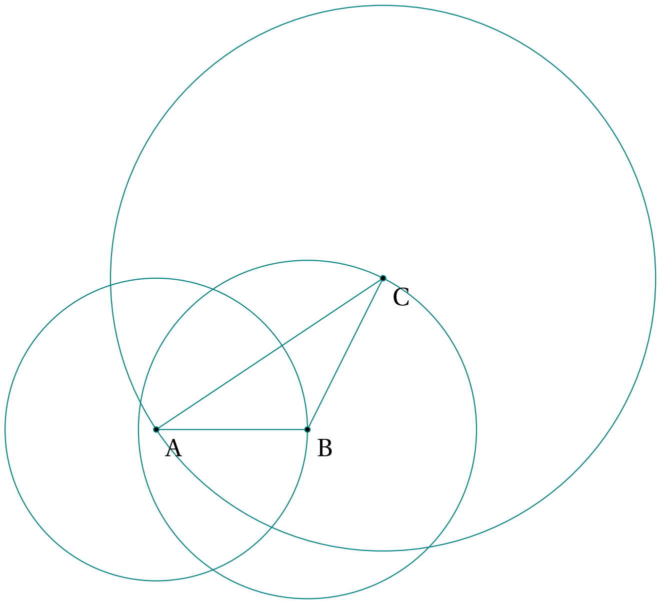
Attention, the arguments are lists of two points. The circles that can be drawn are the same as in the previous macro. An additional option `R` is available to give a measure directly.

arguments	example	explication
(⟨pt1,pt2 pt3,pt4 ...⟩)	(⟨A,B C,D⟩)	List of two points

options	default	definition
through	through	circle with two points defining a radius
diameter	through	circle with two points defining a diameter
R	through	circle characterized by a point and the measurement of a radius

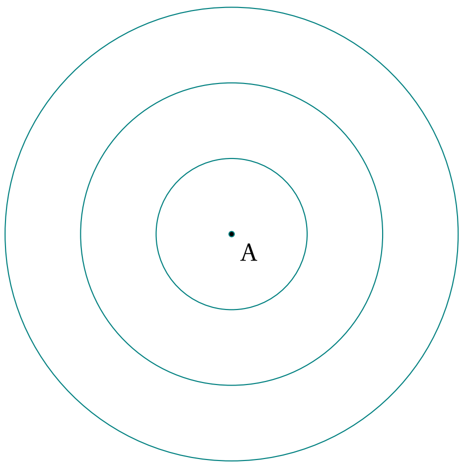
Of course, you have to add all the styles of TikZ for the tracings...

26.2.1. Circles defined by a triangle.



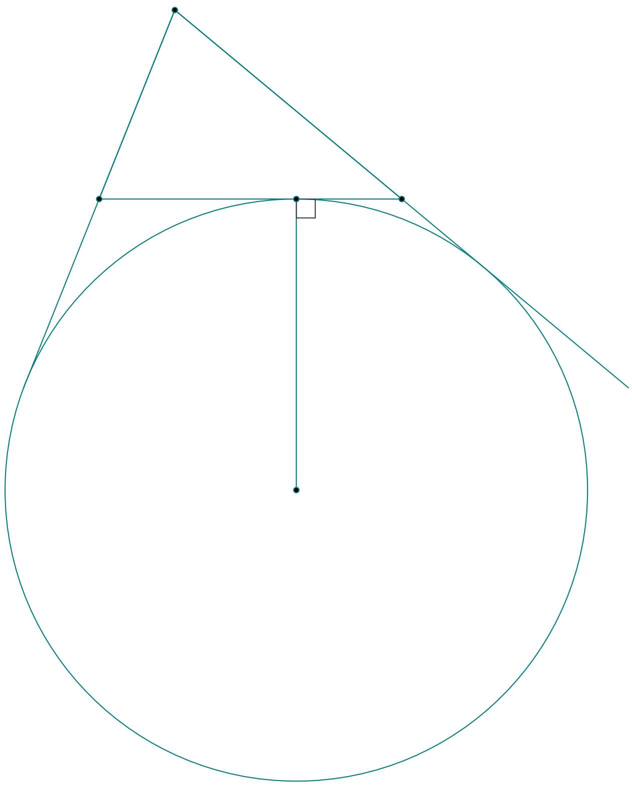

```
\begin{tikzpicture}
  \tkzDefPoints{0/0/A,2/0/B,3/2/C}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawCircles(A,B B,C C,A)
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(A,B,C)
\end{tikzpicture}
```

26.2.2. Concentric circles.




```
\begin{tikzpicture}
   \tkzDefPoint(0,0){A}
   \tkzDrawCircles[R](A,1 A,2 A,3)
   \tkzDrawPoint(A)
   \tkzLabelPoints(A)
\end{tikzpicture}
```

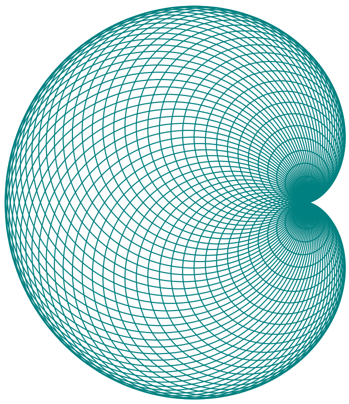
26.2.3. Exinscribed circles.



```
\begin{tikzpicture}[scale=1]
\tkzDefPoints{0/0/A,4/0/B,1/2.5/C}
\tkzDrawPolygon(A,B,C)
\tkzDefCircle[ex](B,C,A)
\tkzGetPoint{J_c} \tkzGetSecondPoint{T_c}
\tkzGetLength{rJc}
\tkzDrawCircle[R](J_c,{\rJc pt})
\tkzDrawLines[add=0 and 1](C,A C,B)
\tkzDrawSegment(J_c,T_c)
\tkzMarkRightAngle(J_c,T_c,B)
\tkzDrawPoints(A,B,C,J_c,T_c)
\end{tikzpicture}
```

26.2.4. Cardioid

Based on an idea by O. Reboux made with pst-eucl (Pstricks module) by D. Rodriguez.
Its name comes from the Greek *kardia (heart)*, in reference to its shape, and was given to it by Johan Castillon (Wikipedia).



```
\begin{tikzpicture}[scale=.5]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(2,0){A}
  \foreach \ang in {5,10,...,360}{%
     \tkzDefPoint(\ang:2){M}
     \tkzDrawCircle(M,A)
   }
\end{tikzpicture}
```

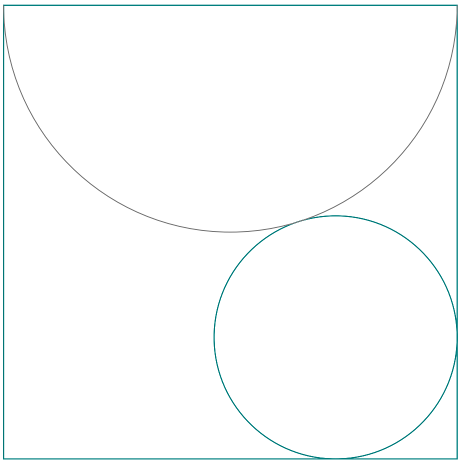
26.3. Drawing semicircle

\tkzDrawSemiCircle[⟨local options⟩](⟨A,B⟩)

arguments	example	explication
(⟨pt1,pt2⟩)	(⟨O,A⟩) or(⟨A,B⟩)	radius or diameter

options	default	definition
through	through	circle characterized by two points defining a radius
diameter	through	circle characterized by two points defining a diameter

26.3.1. Use of \tkzDrawSemiCircle



```
\begin{tikzpicture}
   \tkzDefPoint(0,0){A} \tkzDefPoint(6,0){B}
   \tkzDefSquare(A,B)    \tkzGetPoints{C}{D}
   \tkzDrawPolygon(B,C,D,A)
   \tkzDefPoint(3,6){F}
   \tkzDefTriangle[equilateral](C,D)
   \tkzGetPoint{I}
   \tkzDefPointBy[projection=onto B--C](I)
   \tkzGetPoint{J}
   \tkzInterLL(D,B)(I,J)  \tkzGetPoint{K}
   \tkzDefPointBy[symmetry=center K](B)
   \tkzGetPoint{M}
   \tkzDrawCircle(M,I)
   \tkzCalcLength(M,I) \tkzGetLength{dMI}
   \tkzDrawPolygon(A,B,C,D)
   \tkzDrawCircle[R](M,\dMI)
   \tkzDrawSemiCircle(F,D)
\end{tikzpicture}
```

26.4. Drawing semicircles

\tkzDrawSemiCircles[⟨local options⟩](⟨A,B C,D …⟩)

arguments	example	explication
(⟨pt1,pt2 pt3,pt4 ...⟩)	(⟨A,B C,D⟩)	List of two points

options	default	definition
through	through	circle with two points defining a radius
diameter	through	circle with two points defining a diameter

27. Drawing arcs

```
\tkzDrawArc[⟨local options⟩](⟨O,...⟩)(⟨...⟩)
```

This macro traces the arc of center O. Depending on the options, the arguments differ. It is a question of determining a starting point and an end point. Either the starting point is given, which is the simplest, or the radius of the arc is given. In the latter case, it is necessary to have two angles. Either the angles can be given directly, or nodes associated with the center can be given to determine them. The angles are in degrees.

options	default	definition
towards	towards	O is the center and the arc from A to (OB)
rotate	towards	the arc starts from A and the angle determines its length
R	towards	We give the radius and two angles
R with nodes	towards	We give the radius and two points
angles	towards	We give the radius and two points
delta	0	angle added on each side

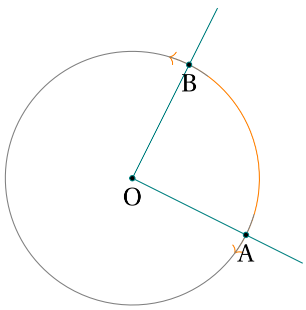
Of course, you have to add all the styles of Ti*k*Z for the tracings...

options	arguments	example
towards	(⟨pt,pt⟩)(⟨pt⟩)	`\tkzDrawArc[delta=10](O,A)(B)`
rotate	(⟨pt,pt⟩)(⟨an⟩)	`\tkzDrawArc[rotate,color=red](O,A)(90)`
R	(⟨pt,r⟩)(⟨an,an⟩)	`\tkzDrawArc[R](O,2)(30,90)`
R with nodes	(⟨pt,r⟩)(⟨pt,pt⟩)	`\tkzDrawArc[R with nodes](O,2)(A,B)`
angles	(⟨pt,pt⟩)(⟨an,an⟩)	`\tkzDrawArc[angles](O,A)(0,90)`

Here are a few examples:

27.1. Option towards

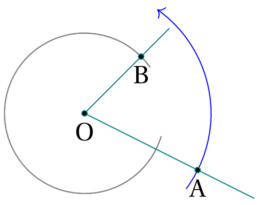
It's useless to put **towards**. In this first example the arc starts from A and goes to B. The arc going from B to A is different. The salient is obtained by going in the direct direction of the trigonometric circle.




```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(2,-1){A}
  \tkzDefPointBy[rotation= center O angle 90](A)
  \tkzGetPoint{B}
  \tkzDrawArc[color=orange,<->](O,A)(B)
  \tkzDrawArc(O,B)(A)
  \tkzDrawLines[add = 0 and .5](O,A O,B)
  \tkzDrawPoints(O,A,B)
  \tkzLabelPoints[below](O,A,B)
\end{tikzpicture}
```

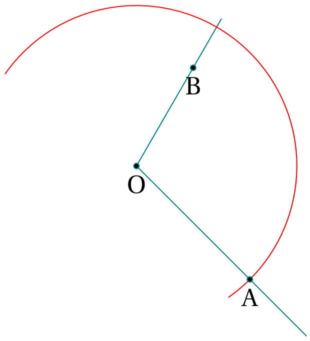
27.2. Option towards

In this one, the arc starts from A but stops on the right (OB).



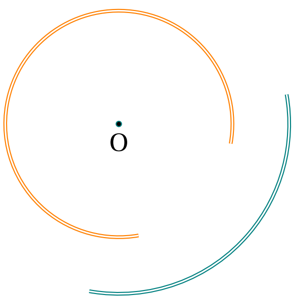

```
\begin{tikzpicture}[scale=0.75]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(2,-1){A}
  \tkzDefPoint(1,1){B}
  \tkzDrawArc[color=blue,->](O,A)(B)
  \tkzDrawArc[color=gray](O,B)(A)
  \tkzDrawArc(O,B)(A)
  \tkzDrawLines[add = 0 and .5](O,A O,B)
  \tkzDrawPoints(O,A,B)
  \tkzLabelPoints[below](O,A,B)
\end{tikzpicture}
```

27.3. Option rotate



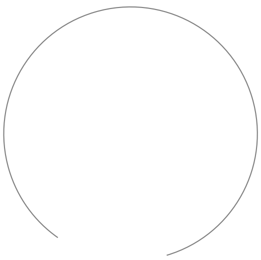

```
\begin{tikzpicture}[scale=0.75]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(2,-2){A}
  \tkzDefPoint(60:2){B}
  \tkzDrawLines[add = 0 and .5](O,A O,B)
  \tkzDrawArc[rotate,color=red](O,A)(180)
  \tkzDrawPoints(O,A,B)
  \tkzLabelPoints[below](O,A,B)
\end{tikzpicture}
```

27.4. Option R




```
\begin{tikzpicture}[scale=0.75]
  \tkzDefPoints{0/0/O}
  \tkzSetUpCompass[<->]
  \tkzDrawArc[R,color=teal,double](O,3)(270,360)
  \tkzDrawArc[R,color=orange,double](O,2)(0,270)
  \tkzDrawPoint(O)
  \tkzLabelPoint[below](O){$O$}
\end{tikzpicture}
```

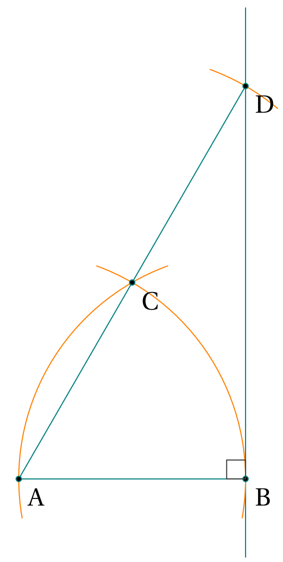
27.5. Option R with nodes



```
\begin{tikzpicture}[scale=0.75]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(2,-1){A}
  \tkzDefPoint(1,1){B}
  \tkzCalcLength(B,A)\tkzGetLength{radius}
  \tkzDrawArc[R with nodes](B,\radius)(A,O)
\end{tikzpicture}
```

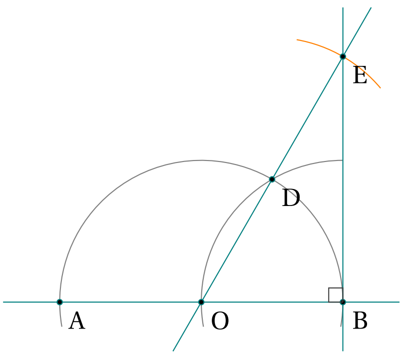
27.6. Option delta

This option allows a bit like `\tkzCompass` to place an arc and overflow on either side. delta is a measure in degrees.



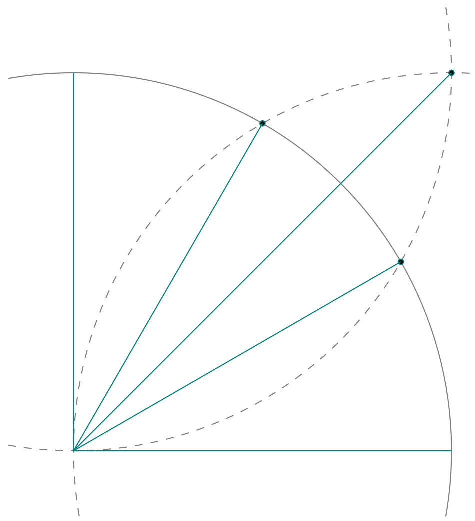

```
\begin{tikzpicture}
 \tkzDefPoint(0,0){A}
 \tkzDefPoint(3,0){B}
 \tkzDefPointBy[rotation= center A angle 60](B)
 \tkzGetPoint{C}
 \begin{scope}% style only local
   \tkzDefPointBy[symmetry= center C](A)
   \tkzGetPoint{D}
   \tkzDrawSegments(A,B A,D)
   \tkzDrawLine(B,D)
   \tkzSetUpCompass[color=orange]
   \tkzDrawArc[orange,delta=10](A,B)(C)
   \tkzDrawArc[orange,delta=10](B,C)(A)
   \tkzDrawArc[orange,delta=10](C,D)(D)
 \end{scope}

 \tkzDrawPoints(A,B,C,D)
 \tkzLabelPoints(A,B,C,D)
 \tkzMarkRightAngle(D,B,A)
\end{tikzpicture}
```

27.7. Option angles: example 1




```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(5,0){B}
  \tkzDefPoint(2.5,0){O}
  \tkzDefPointBy[rotation=center O angle 60](B)
  \tkzGetPoint{D}
  \tkzDefPointBy[symmetry=center D](O)
  \tkzGetPoint{E}
  \begin{scope}
    \tkzDrawArc[angles](O,B)(0,180)
    \tkzDrawArc[angles,](B,O)(100,180)
    \tkzCompass[delta=20](D,E)
    \tkzDrawLines(A,B O,E B,E)
    \tkzDrawPoints(A,B,O,D,E)
  \end{scope}
  \tkzLabelPoints(A,B,O,D,E)
  \tkzMarkRightAngle(O,B,E)
\end{tikzpicture}
```

27.8. Option angles: example 2



```
\begin{tikzpicture}
 \tkzDefPoint(0,0){O}
 \tkzDefPoint(5,0){I}
 \tkzDefPoint(0,5){J}
 \tkzInterCC(O,I)(I,O)\tkzGetPoints{B}{C}
 \tkzInterCC(O,I)(J,O)\tkzGetPoints{D}{A}
 \tkzInterCC(I,O)(J,O)\tkzGetPoints{L}{K}
 \tkzDrawArc[angles](O,I)(0,90)
 \tkzDrawArc[angles,color=gray,
             style=dashed](I,O)(90,180)
 \tkzDrawArc[angles,color=gray,
             style=dashed](J,O)(-90,0)
 \tkzDrawPoints(A,B,K)
 \foreach \point in {I,A,B,J,K}{%
             \tkzDrawSegment(O,\point)}
\end{tikzpicture}
```

28. Drawing a sector or sectors

28.1. \tkzDrawSector

Attention the arguments vary according to the options.

`\tkzDrawSector[⟨local options⟩](⟨O,...⟩)(⟨...⟩)`

options	default	definition
towards	towards	O is the center and the arc from A to (OB)
rotate	towards	the arc starts from A and the angle determines its length
R	towards	We give the radius and two angles
R with nodes	towards	We give the radius and two points

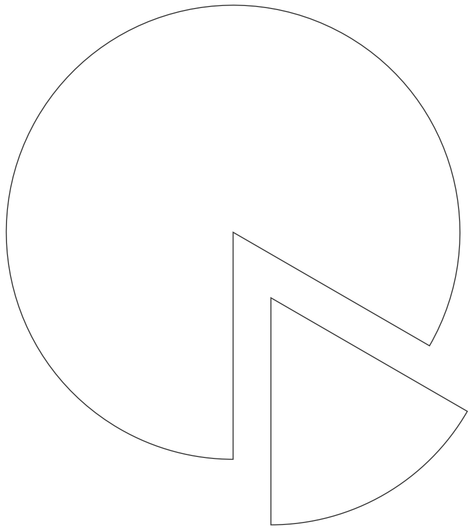
You have to add, of course, all the styles of TikZ for tracings...

options	arguments	example
towards	(⟨pt,pt⟩)(⟨pt⟩)	`\tkzDrawSector(O,A)(B)`
rotate	(⟨pt,pt⟩)(⟨an⟩)	`\tkzDrawSector[rotate,color=red](O,A)(90)`
R	(⟨pt,r⟩)(⟨an,an⟩)	`\tkzDrawSector[R,color=teal](O,2)(30,90)`
R with nodes	(⟨pt,r⟩)(⟨pt,pt⟩)	`\tkzDrawSector[R with nodes](O,2)(A,B)`

Here are a few examples:

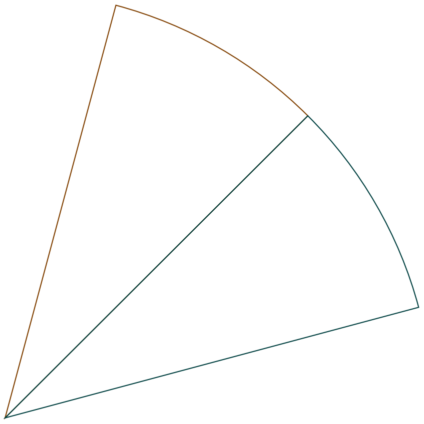
28.1.1. \tkzDrawSector and towards

There's no need to put **towards**. You can use **fill** as an option.



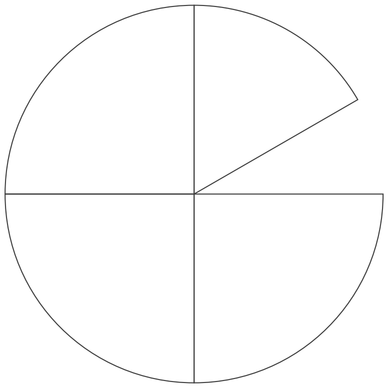
```
\begin{tikzpicture}
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(-30:3){A}
  \tkzDefPointBy[rotation = center O angle -60](A)
  \tkzDrawSector(O,A)(tkzPointResult)
 \begin{scope}[shift={(-60:1)}]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(-30:3){A}
  \tkzDefPointBy[rotation = center O angle -60](A)
  \tkzDrawSector(O,tkzPointResult)(A)
  \end{scope}
\end{tikzpicture}
```

28.1.2. \tkzDrawSector and rotate



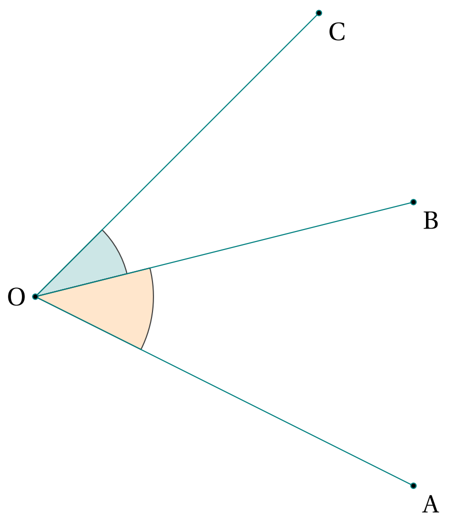
```
\begin{tikzpicture}[scale=2]
 \tkzDefPoint(0,0){O}
 \tkzDefPoint(2,2){A}
 \tkzDrawSector[rotate,draw=orange!50!black](O,A)(30)
 \tkzDrawSector[rotate,draw=teal!50!black](O,A)(-30)
\end{tikzpicture}
```

28.1.3. \tkzDrawSector and R



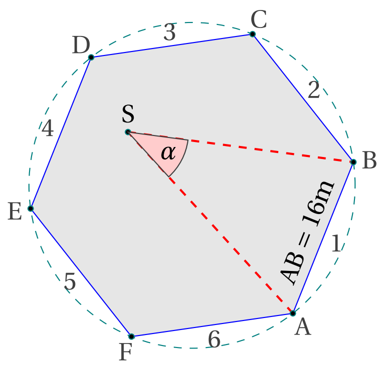
```
\begin{tikzpicture}[scale=1.25]
 \tkzDefPoint(0,0){O}
 \tkzDefPoint(2,-1){A}
 \tkzDrawSector[R](O,2)(30,90)
 \tkzDrawSector[R](O,2)(90,180)
 \tkzDrawSector[R](O,2)(180,270)
 \tkzDrawSector[R](O,2)(270,360)
\end{tikzpicture}
```

28.1.4. \tkzDrawSector and R




```
\begin{tikzpicture}[scale=1.25]
 \tkzDefPoint(0,0){O}
 \tkzDefPoint(4,-2){A}
 \tkzDefPoint(4,1){B}
 \tkzDefPoint(3,3){C}
 \tkzDrawSector[R with nodes,%
                fill=teal!20](O,1)(B,C)
 \tkzDrawSector[R with nodes,%
                fill=orange!20](O,1.25)(A,B)
\tkzDrawSegments(O,A O,B O,C)
\tkzDrawPoints(O,A,B,C)
\tkzLabelPoints(A,B,C)
\tkzLabelPoints[left](O)
\end{tikzpicture}
```

28.1.5. \tkzDrawSector and R with nodes




```
\begin{tikzpicture} [scale=.4]
 \tkzDefPoints{-1/-2/A,1/3/B}
 \tkzDefRegPolygon[side,sides=6](A,B)
 \tkzGetPoint{O}
 \tkzDrawPolygon[fill=black!10, draw=blue](P1,P...,P6)
 \tkzLabelRegPolygon[sep=1.05](O){A,...,F}
 \tkzDrawCircle[dashed](O,A)
 \tkzLabelSegment[above,sloped,
                  midway](A,B){\(A B = 16m\)}
 \foreach \i  [count=\xi from 1]  in {2,...,6,1}
   {%
    \tkzDefMidPoint(P\xi,P\i)
    \path (O) to [pos=1.1] node {\xi} (tkzPointResult) ;
    }
  \tkzDefRandPointOn[segment = P3--P5]
  \tkzGetPoint{S}
  \tkzDrawSegments[thick,dashed,red](A,S S,B)
  \tkzDrawPoints(P1,P...,P6,S)
  \tkzLabelPoint[left,above](S){$S$}
  \tkzDrawSector[R with nodes,fill=red!20](S,2)(A,B)
  \tkzLabelAngle[pos=1.5](A,S,B){$\alpha$}
\end{tikzpicture}
```

28.2. Coloring a disc

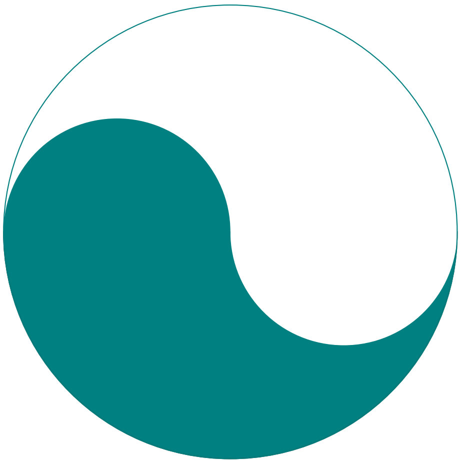
This was possible with the macro `\tkzDrawCircle`, but disk tracing was mandatory, this is no longer the case.

`\tkzFillCircle[⟨local options⟩](⟨A,B⟩)`

options	default	definition
radius	radius	two points define a radius
R	radius	a point and the measurement of a radius

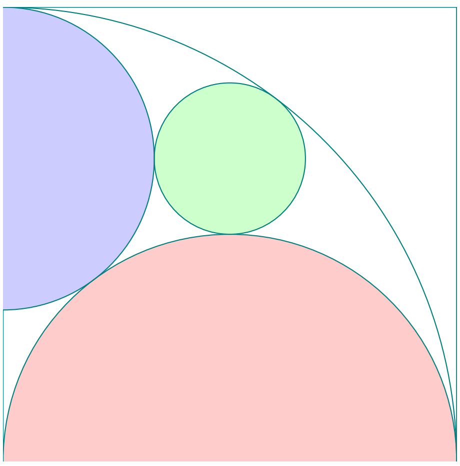
You don't need to put **radius** because that's the default option. Of course, you have to add all the styles of TikZ for the plots.

28.2.1. Yin and Yang



```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(-4,0){A}
  \tkzDefPoint(4,0){B}
  \tkzDefPoint(-2,0){I}
  \tkzDefPoint(2,0){J}
  \tkzDrawSector[fill=teal](O,A)(B)
  \tkzFillCircle[fill=white](J,B)
  \tkzFillCircle[fill=teal](I,A)
  \tkzDrawCircle(O,A)
\end{tikzpicture}
```

28.2.2. From a sangaku

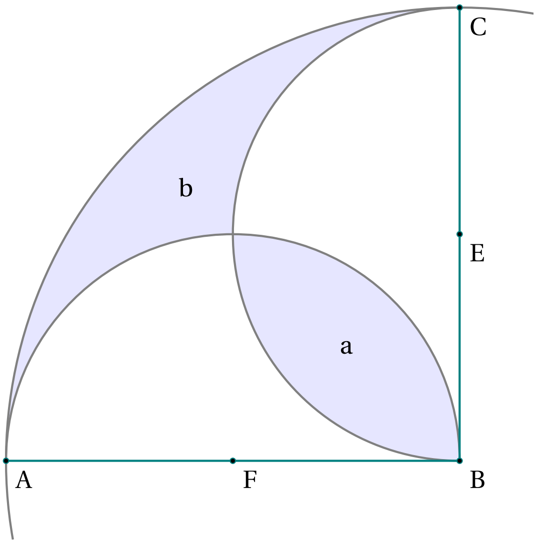


```
\begin{tikzpicture}
   \tkzDefPoint(0,0){B}  \tkzDefPoint(6,0){C}%
   \tkzDefSquare(B,C)    \tkzGetPoints{D}{A}
   \tkzClipPolygon(B,C,D,A)
   \tkzDefMidPoint(A,D)  \tkzGetPoint{F}
   \tkzDefMidPoint(B,C)  \tkzGetPoint{E}
   \tkzDefMidPoint(B,D)  \tkzGetPoint{Q}
   \tkzDefTangent[from = B](F,A) \tkzGetPoints{G}{H}
   \tkzInterLL(F,G)(C,D) \tkzGetPoint{J}
   \tkzInterLL(A,J)(F,E) \tkzGetPoint{K}
   \tkzDefPointBy[projection=onto B--A](K)
   \tkzGetPoint{M}
   \tkzDrawPolygon(A,B,C,D)
   \tkzFillCircle[red!20](E,B)
   \tkzFillCircle[blue!20](M,A)
   \tkzFillCircle[green!20](K,Q)
  \tkzDrawCircles(B,A M,A E,B K,Q)
\end{tikzpicture}
```

28.2.3. Clipping and filling part I

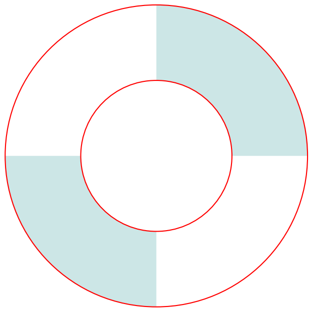

```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,4/0/B,2/2/O,3/4/X,4/1/Y,1/0/Z,
              0/3/W,3/0/R,4/3/S,1/4/T,0/1/U}
\tkzDefSquare(A,B)\tkzGetPoints{C}{D}
\tkzDefPointWith[colinear normed=at X,K=1](O,X)
 \tkzGetPoint{F}
\begin{scope}
  \tkzFillCircle[fill=teal!20](O,F)
  \tkzFillPolygon[white](A,...,D)
  \tkzClipPolygon(A,...,D)
  \foreach \c/\t in {S/C,R/B,U/A,T/D}
  {\tkzFillCircle[teal!20](\c,\t)}
\end{scope}
\foreach \c/\t in {X/C,Y/B,Z/A,W/D}
{\tkzFillCircle[white](\c,\t)}
  \foreach \c/\t in {S/C,R/B,U/A,T/D}
  {\tkzFillCircle[teal!20](\c,\t)}
\end{tikzpicture}
```

28.2.4. Clipping and filling part II




```
\begin{tikzpicture}[scale=.75]
\tkzDefPoints{0/0/A,8/0/B,8/8/C,0/8/D}
\tkzDefMidPoint(A,B) \tkzGetPoint{F}
\tkzDefMidPoint(B,C) \tkzGetPoint{E}
\tkzDefMidPoint(D,B) \tkzGetPoint{I}
\tkzDefMidPoint(I,B) \tkzGetPoint{a}
\tkzInterLC(B,I)(B,C) \tkzGetSecondPoint{K}
\tkzDefMidPoint(I,K) \tkzGetPoint{b}
\begin{scope}
  \tkzFillSector[fill=blue!10](B,C)(A)
  \tkzDrawSemiCircle[diameter,fill=white](A,B)
  \tkzDrawSemiCircle[diameter,fill=white](B,C)
  \tkzClipCircle(E,B)
  \tkzClipCircle(F,B)
  \tkzFillCircle[fill=blue!10](B,A)
\end{scope}
\tkzDrawSemiCircle[thick](F,B)
\tkzDrawSemiCircle[thick](E,C)
\tkzDrawArc[thick](B,C)(A)
\tkzDrawSegments[thick](A,B B,C)
\tkzDrawPoints(A,B,C,E,F)
\tkzLabelPoints[centered](a,b)
\tkzLabelPoints(A,B,C,E,F)
\end{tikzpicture}
```

28.2.5. Clipping and filling part III



```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A} \tkzDefPoint(1,0){B}
  \tkzDefPoint(2,0){C} \tkzDefPoint(-3,0){a}
  \tkzDefPoint(3,0){b}  \tkzDefPoint(0,3){c}
  \tkzDefPoint(0,-3){d}
\begin{scope}
 \tkzClipPolygon(a,b,c,d)
 \tkzFillCircle[teal!20](A,C)
\end{scope}
 \tkzFillCircle[white](A,B)
 \tkzDrawCircle[color=red](A,C)
 \tkzDrawCircle[color=red](A,B)
\end{tikzpicture}
```

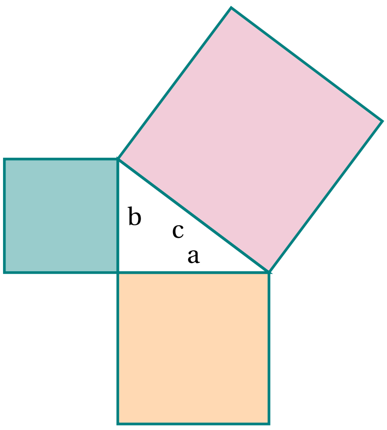
28.3. Coloring a polygon

\tkzFillPolygon[⟨local options⟩](⟨points list⟩)

You can color by drawing the polygon, but in this case you color the inside of the polygon without drawing it.

arguments	example	explication
(⟨pt1,pt2,…⟩)	(⟨A,B,…⟩)	

28.3.1. \tkzFillPolygon




```
\begin{tikzpicture}[scale=.5]
   \tkzDefPoint(0,0){C} \tkzDefPoint(4,0){A}
   \tkzDefPoint(0,3){B}
   \tkzDefSquare(B,A)      \tkzGetPoints{E}{F}
   \tkzDefSquare(A,C)      \tkzGetPoints{G}{H}
   \tkzDefSquare(C,B)       \tkzGetPoints{I}{J}
   \tkzFillPolygon[color  =  orange!30   ](A,C,G,H)
   \tkzFillPolygon[color  =  teal!40  ](C,B,I,J)
   \tkzFillPolygon[color  =  purple!20](B,A,E,F)
   \tkzDrawPolygon[line width  =  1pt](A,B,C)
   \tkzDrawPolygon[line width  =  1pt](A,C,G,H)
   \tkzDrawPolygon[line width  =  1pt](C,B,I,J)
   \tkzDrawPolygon[line width  =  1pt](B,A,E,F)
   \tkzLabelSegment[above](C,A){$a$}
   \tkzLabelSegment[right](B,C){$b$}
   \tkzLabelSegment[below left](B,A){$c$}
\end{tikzpicture}
```

28.4. \tkzFillSector

Attention the arguments vary according to the options.

`\tkzFillSector[⟨local options⟩](⟨O,...⟩)(⟨...⟩)`

options	default	definition
towards	towards	O is the center and the arc from A to (OB)
rotate	towards	the arc starts from A and the angle determines its length
R	towards	We give the radius and two angles
R with nodes	towards	We give the radius and two points

Of course, you have to add all the styles of TikZ for the tracings...

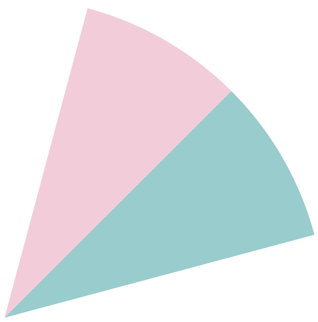
options	arguments	example
towards	(⟨pt,pt⟩)(⟨pt⟩)	`\tkzFillSector(O,A)(B)`
rotate	(⟨pt,pt⟩)(⟨an⟩)	`\tkzFillSector[rotate,color=red](O,A)(90)`
R	(⟨pt,r⟩)(⟨an,an⟩)	`\tkzFillSector[R,color=blue](O,2)(30,90)`
R with nodes	(⟨pt,r⟩)(⟨pt,pt⟩)	`\tkzFillSector[R with nodes](O,2)(A,B)`

28.4.1. \tkzFillSector and towards

It is useless to put **towards** and you will notice that the contours are not drawn, only the surface is colored.

```
\begin{tikzpicture}[scale=.6]
\tkzDefPoint(0,0){O}
\tkzDefPoint(-30:3){A}
\tkzDefPointBy[rotation = center O angle -60](A)
\tkzFillSector[fill=purple!20](O,A)(tkzPointResult)
  \begin{scope}[shift={(-60:1)}]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(-30:3){A}
  \tkzDefPointBy[rotation = center O angle -60](A)
  \tkzGetPoint{A'}
  \tkzFillSector[color=teal!40](O,A')(A)
    \end{scope}
\end{tikzpicture}
```

28.4.2. \tkzFillSector and rotate



```
\begin{tikzpicture}[scale=1.5]
 \tkzDefPoint(0,0){O} \tkzDefPoint(2,2){A}
 \tkzFillSector[rotate,color=purple!20](O,A)(30)
 \tkzFillSector[rotate,color=teal!40](O,A)(-30)
\end{tikzpicture}
```

28.5. Colour an angle: \tkzFillAngle

The simplest operation

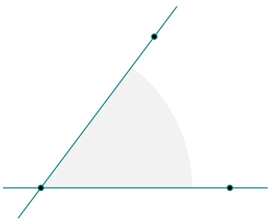
`\tkzFillAngle[⟨local options⟩](⟨A,O,B⟩)`

O is the vertex of the angle. OA and OB are the sides. Attention the angle is determined by the order of the points.

options	default	definition
size	1	this option determines the radius of the coloured angular sector.

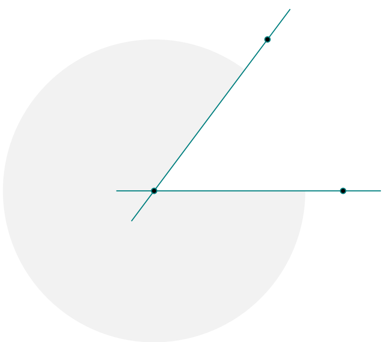
Of course, you have to add all the styles of TikZ, like the use of fill and shade...

28.5.1. Example with size

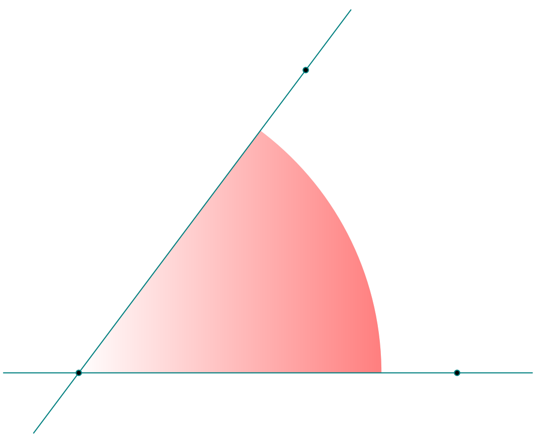


```
\begin{tikzpicture}
   \tkzInit
   \tkzDefPoints{0/0/O,2.5/0/A,1.5/2/B}
   \tkzFillAngle[size=2, fill=gray!10](A,O,B)
   \tkzDrawLines(O,A O,B)
   \tkzDrawPoints(O,A,B)
\end{tikzpicture}
```

28.5.2. Changing the order of items



```
\begin{tikzpicture}
   \tkzInit
   \tkzDefPoints{0/0/O,2.5/0/A,1.5/2/B}
   \tkzFillAngle[size=2,fill=gray!10](B,O,A)
   \tkzDrawLines(O,A O,B)
   \tkzDrawPoints(O,A,B)
\end{tikzpicture}
```

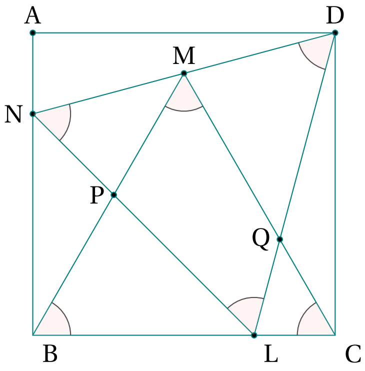


```
\begin{tikzpicture}
   \tkzInit
   \tkzDefPoints{0/0/O,5/0/A,3/4/B}
   % Don't forget {} to get, () to use
   \tkzFillAngle[size=4,left color=white,
                 right color=red!50](A,O,B)
   \tkzDrawLines(O,A O,B)
   \tkzDrawPoints(O,A,B)
\end{tikzpicture}
```

`\tkzFillAngles[⟨local options⟩](⟨A,O,B⟩)(⟨A',O',B'⟩)etc.`

With common options, there is a macro for multiple angles.

28.5.3. Multiples angles




```
\begin{tikzpicture}[scale=0.5]
  \tkzDefPoints{0/0/B,8/0/C,0/8/A,8/8/D}
  \tkzDrawPolygon(B,C,D,A)
  \tkzDefTriangle[equilateral](B,C) \tkzGetPoint{M}
  \tkzInterLL(D,M)(A,B) \tkzGetPoint{N}
  \tkzDefPointBy[rotation=center N angle -60](D)
  \tkzGetPoint{L}
  \tkzInterLL(N,L)(M,B)     \tkzGetPoint{P}
  \tkzInterLL(M,C)(D,L)     \tkzGetPoint{Q}
  \tkzDrawSegments(D,N N,L L,D B,M M,C)
  \tkzDrawPoints(L,N,P,Q,M,A,D)
  \tkzLabelPoints[left](N,P,Q)
  \tkzLabelPoints[above](M,A,D)
  \tkzLabelPoints(L,B,C)
  \tkzMarkAngles(C,B,M B,M,C M,C,B D,L,N L,N,D N,D,L)
  \tkzFillAngles[fill=red!20,opacity=.2](C,B,M%
      B,M,C M,C,B D,L,N L,N,D N,D,L)
\end{tikzpicture}
```

29. Controlling Bounding Box

From the **PgfManual** :"When you add the clip option, the current path is used for clipping subsequent drawings. Clipping never enlarges the clipping area. Thus, when you clip against a certain path and then clip again against another path, you clip against the intersection of both. The only way to enlarge the clipping path is to end the pgfscope in which the clipping was done. At the end of a pgfscope the clipping path that was in force at the beginning of the scope is reinstalled."
First of all, you don't have to deal with Ti*k*Z the size of the bounding box. Early versions of `tkz-euclide` did not control the size of the bounding box, now with `tkz-euclide` 4 the size of the bounding box is limited.
The initial bounding box after using the macro `\tkzInit` is defined by the rectangle based on the points $(0,0)$ and $(10,10)$. The `\tkzInit` macro allows this initial bounding box to be modified using the arguments (**`xmin`**, **`xmax`**, **`ymin`**, and **`ymax`**). Of course any external trace modifies the bounding box. Ti*k*Z maintains that bounding box. It is possible to influence this behavior either directly with commands or options in Ti*k*Z such as a command like `\useasboundingbox` or the option **`use as bounding box`**. A possible consequence is to reserve a box for a figure but the figure may overflow the box and spread over the main text. The following command `\pgfresetboundingbox` clears a bounding box and establishes a new one.

29.1. Utility of \tkzInit

However, it is sometimes necessary to control the size of what will be displayed. To do this, you need to have prepared the bounding box you are going to work in, this is the role of the macro `\tkzInit`. For some drawings, it is interesting to fix the extreme values (xmin,xmax,ymin and ymax) and to "clip" the definition rectangle in order to control the size of the figure as well as possible.
The two macros that are useful for controlling the bounding box:

– `\tkzInit`

– `\tkzClip`

To this, I added macros directly linked to the bounding box. You can now view it, backup it, restore it (see the section Bounding Box).

29.2. \tkzInit

`\tkzInit[⟨local options⟩]`

options	default	definition
xmin	0	minimum value of the abscissae in cm
xmax	10	maximum value of the abscissae in cm
xstep	1	difference between two graduations in x
ymin	0	minimum y-axis value in cm
ymax	10	maximum y-axis value in cm
ystep	1	difference between two graduations in y

The role of `\tkzInit` is to define a orthogonal coordinates system and a rectangular part of the plane in which you will place your drawings using Cartesian coordinates. This macro allows you to define your working environment as with a calculator. With **`tkz-euclide`** 4 `\xstep` and `\ystep` are always 1. Logically it is no longer useful to use `\tkzInit`, except for an action like "Clipping Out".

29.3. \tkzClip

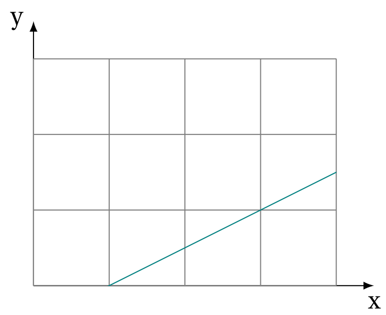
`\tkzClip[⟨local options⟩]`

The role of this macro is to make invisible what is outside the rectangle defined by (xmin ; ymin) and (xmax ; ymax).

options	default	definition
space	1	added value on the right, left, bottom and top of the background

The role of the **space** option is to enlarge the visible part of the drawing. This part becomes the rectangle defined by (xmin-space ; ymin-space) and (xmax+space ; ymax+space). **space** can be negative! The unit is cm and should not be specified.

The role of this macro is to "clip" the initial rectangle so that only the paths contained in this rectangle are drawn.



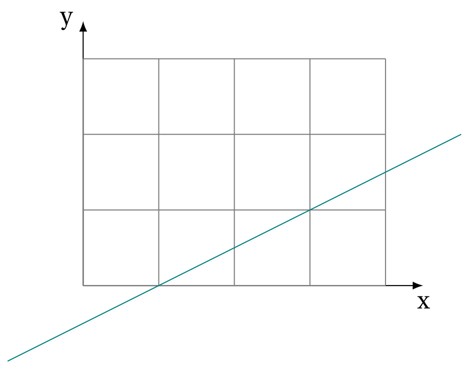

```
\begin{tikzpicture}
 \tkzInit[xmax=4, ymax=3]
 \tkzDefPoints{-1/-1/A,5/2/B}
 \tkzDrawX \tkzDrawY
 \tkzGrid
 \tkzClip
 \tkzDrawSegment(A,B)
\end{tikzpicture}
```

It is possible to add a bit of space

```
\tkzClip[space=1]
```

29.4. \tkzClip and the option space

This option allows you to add some space around the "clipped" rectangle.




```
\begin{tikzpicture}
 \tkzInit[xmax=4, ymax=3]
 \tkzDefPoints{-1/-1/A,5/2/B}
 \tkzDrawX \tkzDrawY
 \tkzGrid
 \tkzClip[space=1]
 \tkzDrawSegment(A,B)
\end{tikzpicture}
```

The dimensions of the "clipped" rectangle are **xmin-1**, **ymin-1**, **xmax+1** and **ymax+1**.

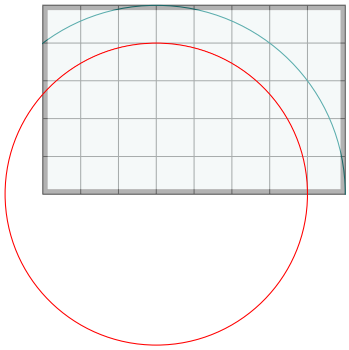
29.5. tkzShowBB

The simplest macro.

`\tkzShowBB[⟨local options⟩]`

This macro displays the bounding box. A rectangular frame surrounds the bounding box. This macro accepts Ti*k*Z options.

29.5.1. Example with \tkzShowBB



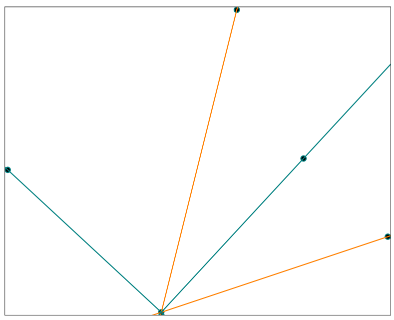
```
\begin{tikzpicture}[scale=.5]
  \tkzInit[ymax=5,xmax=8]
  \tkzGrid
  \tkzDefPoint(3,0){A}
   \begin{scope}
    \tkzClipBB
    \tkzDrawCircle[R](A,5)
     \tkzShowBB[line width = 4pt,fill=teal!10,opacity=.4]
   \end{scope}
\tkzDrawCircle[R,red](A,4)
\end{tikzpicture}
```

29.6. tkzClipBB

`\tkzClipBB`

The idea is to limit future constructions to the current bounding box.

29.6.1. Example with \tkzClipBB and the bisectors



```
\begin{tikzpicture}
\tkzInit[xmin=-3,xmax=6, ymin=-1,ymax=6]
\tkzDefPoint(0,0){O}\tkzDefPoint(3,1){I}
\tkzDefPoint(1,4){J}
\tkzDefLine[bisector](I,O,J) \tkzGetPoint{i}
\tkzDefLine[bisector out](I,O,J) \tkzGetPoint{j}
\tkzDrawPoints(O,I,J,i,j)
\tkzClipBB
\tkzDrawLines[add = 1 and 2,color=orange](O,I O,J)
\tkzDrawLines[add = 1 and 2](O,i O,j)
\tkzShowBB
\end{tikzpicture}
```

30. Clipping different objects

30.1. Clipping a polygon

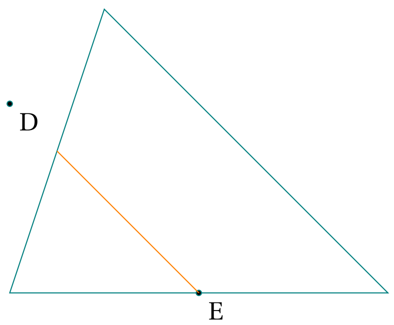
`\tkzClipPolygon[⟨local options⟩](⟨points list⟩)`

This macro makes it possible to contain the different plots in the designated polygon.

arguments	example	explication
(⟨pt1,pt2,pt3,...⟩)	(⟨A,B,C⟩)	

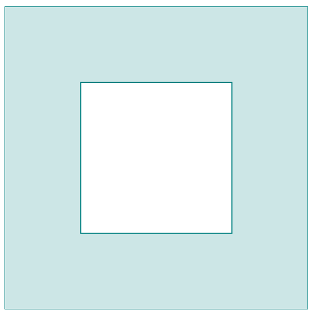
options	default	definition
out		allows to clip the outside of the object

30.1.1. \tkzClipPolygon



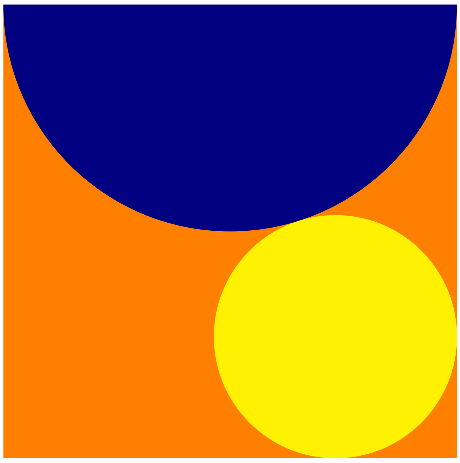

```
  \begin{tikzpicture}[scale=1.25]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(4,0){B}
  \tkzDefPoint(1,3){C}
  \tkzDrawPolygon(A,B,C)
  \tkzDefPoint(0,2){D}
  \tkzDefPoint(2,0){E}
  \tkzDrawPoints(D,E)
  \tkzLabelPoints(D,E)
  \tkzClipPolygon(A,B,C)
  \tkzDrawLine[new](D,E)
\end{tikzpicture}
```

30.1.2. \tkzClipPolygon[out]



```
  \begin{tikzpicture}[scale=1]
  \tkzDefPoint(0,0){P1}
  \tkzDefPoint(4,0){P2}
  \tkzDefPoint(4,4){P3}
  \tkzDefPoint(0,4){P4}
  \tkzDefPoint(1,1){Q1}
  \tkzDefPoint(3,1){Q2}
  \tkzDefPoint(3,3){Q3}
  \tkzDefPoint(1,3){Q4}
  \tkzDrawPolygon(P1,P2,P3,P4)
  \begin{scope}
     \tkzClipPolygon[out](Q1,Q2,Q3,Q4)
    \tkzFillPolygon[teal!20](P1,P2,P3,P4)
  \end{scope}
  \tkzDrawPolygon(Q1,Q2,Q3,Q4)
  \end{tikzpicture}
```

30.1.3. Example: use of "Clip" for Sangaku in a square



```
\begin{tikzpicture}[scale=.75]
 \tkzDefPoint(0,0){A} \tkzDefPoint(8,0){B}
 \tkzDefSquare(A,B)   \tkzGetPoints{C}{D}
 \tkzDefPoint(4,8){F}
 \tkzDefTriangle[equilateral](C,D)
 \tkzGetPoint{I}
 \tkzDefPointBy[projection=onto B--C](I)
 \tkzGetPoint{J}
 \tkzInterLL(D,B)(I,J)  \tkzGetPoint{K}
 \tkzDefPointBy[symmetry=center K](B)
 \tkzGetPoint{M}
 \tkzClipPolygon(B,C,D,A)
 \tkzCalcLength(M,I)   \tkzGetLength{dMI}
 \tkzFillPolygon[color = orange](A,B,C,D)
 \tkzFillCircle[R,color = yellow](M,\dMI)
 \tkzFillCircle[R,color = blue!50!black](F,4)
\end{tikzpicture}
```

30.2. Clipping a disc

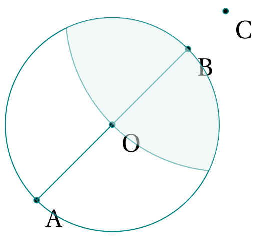
`\tkzClipCircle[⟨local options⟩](⟨A,B⟩) or (⟨A,r⟩)`

arguments	example	explication
(⟨A,B⟩) or (⟨A,r⟩)	(⟨A,B⟩) or (⟨A,2cm⟩)	AB radius or diameter

options	default	definition
radius	radius	circle characterized by two points defining a radius
R	radius	circle characterized by a point and the measurement of a radius
out		allows to clip the outside of the object

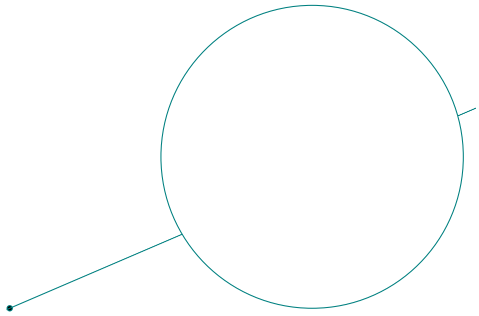
It is not necessary to put **radius** because that is the default option.

30.2.1. Simple clip



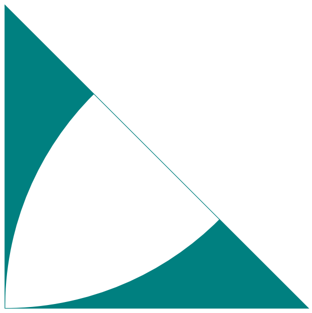

```
\begin{tikzpicture}[scale=.5]
  \tkzDefPoint(0,0){A} \tkzDefPoint(2,2){O}
  \tkzDefPoint(4,4){B} \tkzDefPoint(5,5){C}
  \tkzDrawPoints(O,A,B,C)
  \tkzLabelPoints(O,A,B,C)
  \tkzDrawCircle(O,A)
  \tkzClipCircle(O,A)
  \tkzDrawLine(A,C)
  \tkzDrawCircle[fill=teal!10,opacity=.5](C,O)
\end{tikzpicture}
```

30.3. Clip out



```
\begin{tikzpicture}
  \tkzInit[xmin=-3,ymin=-2,xmax=4,ymax=3]
   \tkzDefPoint(0,0){O}
   \tkzDefPoint(-4,-2){A}
   \tkzDefPoint(3,1){B}
   \tkzDrawCircle[R](O,2)
   \tkzDrawPoints(A,B) % to have a good bounding box
   \begin{scope}
    \tkzClipCircle[out,R](O,2)
    \tkzDrawLines(A,B)
   \end{scope}
\end{tikzpicture}
```

30.4. Intersection of disks



```
\begin{tikzpicture}
\tkzDefPoints{0/0/O,4/0/A,0/4/B}
\tkzDrawPolygon[fill=teal](O,A,B)
\tkzClipPolygon(O,A,B)
\tkzClipCircle(A,O)
\tkzClipCircle(B,O)
\tkzFillPolygon[white](O,A,B)
\end{tikzpicture}
```

see a more complex example about clipping here : 46.6

30.5. Clipping a sector

Attention the arguments vary according to the options.

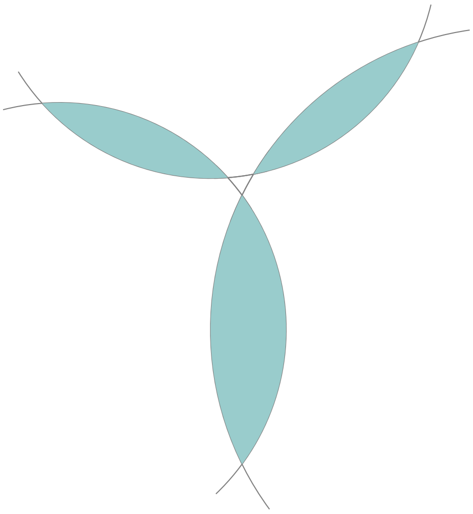
`\tkzClipSector[⟨local options⟩](⟨O,...⟩)(⟨...⟩)`

options	default	definition
towards	towards	O is the centre and the sector starts from A to (OB)
rotate	towards	The sector starts from A and the angle determines its amplitude.
R	towards	We give the radius and two angles

You have to add, of course, all the styles of Ti*k*Z for tracings...

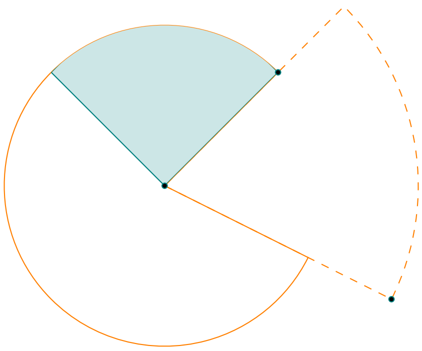
options	arguments	example
towards	(⟨pt,pt⟩)(⟨pt⟩)	`\tkzClipSector(O,A)(B)`
rotate	(⟨pt,pt⟩)(⟨angle⟩)	`\tkzClipSector[rotate](O,A)(90)`
R	(⟨pt,r⟩)(⟨angle 1,angle 2⟩)	`\tkzClipSector[R](O,2)(30,90)`

30.5.1. Example 1



```
\begin{tikzpicture}[scale=0.5]
\tkzDefPoint(0,0){a}
\tkzDefPoint(12,0){b}
\tkzDefPoint(4,10){c}
\tkzInterCC[R](a,6)(b,8)
\tkzGetFirstPoint{AB1} \tkzGetSecondPoint{AB2}
\tkzInterCC[R](a,6)(c,6)
\tkzGetFirstPoint{AC1} \tkzGetSecondPoint{AC2}
\tkzInterCC[R](b,8)(c,6)
\tkzGetFirstPoint{BC1} \tkzGetSecondPoint{BC2}
\tkzDrawArc(a,AB2)(AB1)
\tkzDrawArc(b,AB1)(AB2)
\tkzDrawArc(a,AC2)(AC1)
\tkzDrawArc(c,AC1)(AC2)
\tkzDrawArc(b,BC2)(BC1)
\tkzDrawArc(c,BC1)(BC2)
\begin{scope}
\tkzClipSector(b,BC2)(BC1)
\tkzFillSector[teal!40!white](c,BC1)(BC2)
\end{scope}
\begin{scope}
\tkzClipSector(a,AB2)(AB1)
\tkzFillSector[teal!40!white](b,AB1)(AB2)
\end{scope}
\begin{scope}
\tkzClipSector(a,AC2)(AC1)
\tkzFillSector[teal!40!white](c,AC1)(AC2)
\end{scope}
\end{tikzpicture}
```

30.5.2. Example 2



```
\begin{tikzpicture}[scale=1.5]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(2,-1){A}
  \tkzDefPoint(1,1){B}
  \tkzDrawSector[new,dashed](O,A)(B)
  \tkzDrawSector[new](O,B)(A)
\begin{scope}
\tkzClipSector(O,B)(A)
\tkzDrawSquare[color=teal,fill=teal!20](O,B)
\end{scope}
\tkzDrawPoints(A,B,O)
\end{tikzpicture}
```

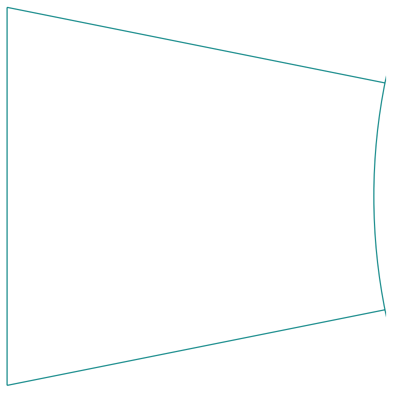
30.6. Options from TikZ: trim left or right

See the **pgfmanual**

30.7. TikZ Controls \pgfinterruptboundingbox and \endpgfinterruptboundingbox

This command temporarily interrupts the calculation of the box and configures a new box. See the **pgfmanual**

30.7.1. Example about contolling the bouding box



```
\begin{tikzpicture}
\tkzDefPoint(0,5){A}\tkzDefPoint(5,4){B}
\tkzDefPoint(0,0){C}\tkzDefPoint(5,1){D}
\tkzDrawSegments(A,B C,D A,C)
\pgfinterruptboundingbox
   \tkzInterLL(A,B)(C,D)\tkzGetPoint{I}
\endpgfinterruptboundingbox
\tkzClipBB
\tkzDrawCircle(I,B)
\end{tikzpicture}
```

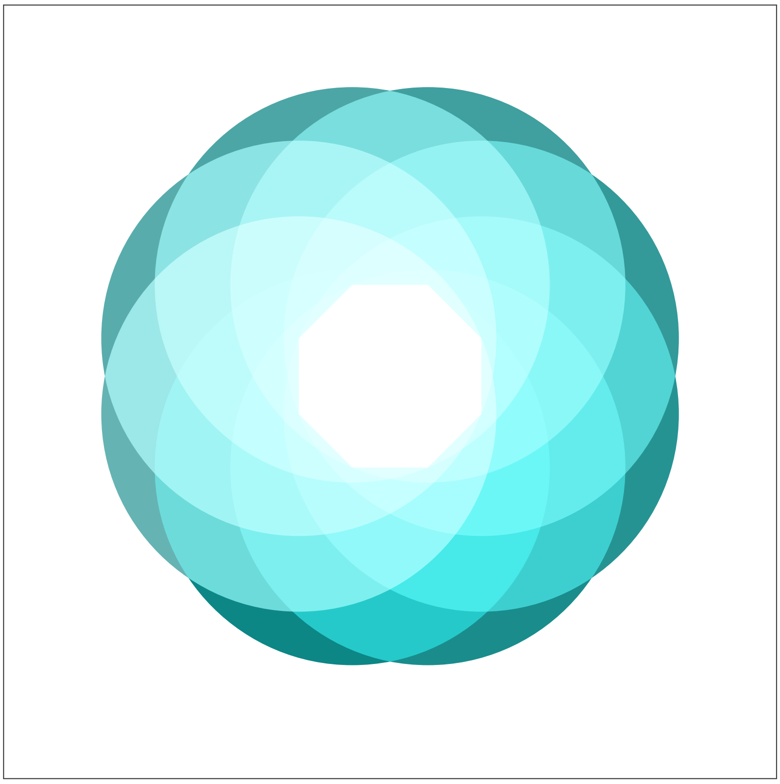
30.8. Reverse clip: tkzreverseclip

In order to use this option, a bounding box must be defined.

```
\tikzset{tkzreverseclip/.style={insert path={
   (current bounding box.south west) --(current bounding box.north west)
 --(current bounding box.north east) --  (current bounding box.south east)
 -- cycle} }}
```

30.8.1. Example with \tkzClipPolygon[out]

`\tkzClipPolygon[out]`, `\tkzClipCircle[out]` use this option.



```
\fbox{\begin{tikzpicture}[scale=1]
\tkzInit[xmin=-5,xmax=5,ymin=-4,ymax=6]
\tkzClip
  \tkzDefPoints{-.5/0/P1,.5/0/P2}
  \foreach \i [count=\j from 3] in {2,...,7}{%
      \tkzDefShiftPoint[P\i]({45*(\i-1)}:1){P\j}}
  \tkzClipPolygon[out](P1,P...,P8)
  \tkzCalcLength(P1,P5)\tkzGetLength{r}
  \begin{scope}[blend group=screen]
     \foreach \i in {1,...,8}{%
       \pgfmathparse{100-5*\i}
       \tkzFillCircle[R,color=teal!%
       \pgfmathresult](P\i,\r)}
    \end{scope}
\end{tikzpicture}}
```

Part V.

Marking

30.9. Mark a segment \tkzMarkSegment

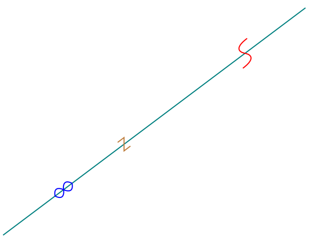
`\tkzMarkSegment[⟨local options⟩](⟨pt1,pt2⟩)`

The macro allows you to place a mark on a segment.

options	default	definition
pos	.5	position of the mark
color	black	color of the mark
mark	none	choice of the mark
size	4pt	size of the mark

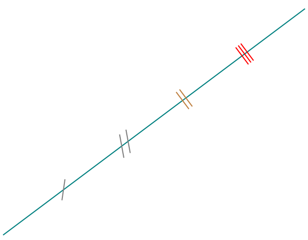
Possible marks are those provided by TikZ, but other marks have been created based on an idea by Yves Combe.

30.9.1. Several marks



```
\begin{tikzpicture}
  \tkzDefPoint(2,1){A}
  \tkzDefPoint(6,4){B}
  \tkzDrawSegment(A,B)
  \tkzMarkSegment[color=brown,size=2pt,pos=0.4, mark=z](A,B)
  \tkzMarkSegment[color=blue,pos=0.2, mark=oo](A,B)
  \tkzMarkSegment[pos=0.8,mark=s,color=red](A,B)
\end{tikzpicture}
```

30.9.2. Use of mark



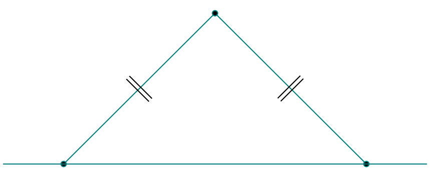
```
\begin{tikzpicture}
  \tkzDefPoint(2,1){A}
  \tkzDefPoint(6,4){B}
  \tkzDrawSegment(A,B)
  \tkzMarkSegment[color=gray,pos=0.2,mark=s|](A,B)
  \tkzMarkSegment[color=gray,pos=0.4,mark=s||](A,B)
  \tkzMarkSegment[color=brown,pos=0.6,mark=||](A,B)
  \tkzMarkSegment[color=red,pos=0.8,mark=|||](A,B)
\end{tikzpicture}
```

30.10. Marking segments \tkzMarkSegments

`\tkzMarkSegments[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩)`

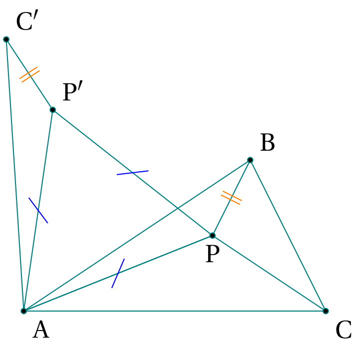
Arguments are a list of pairs of points separated by spaces. The styles of TikZ are available for plots.

30.10.1. Marks for an isosceles triangle



```
\begin{tikzpicture}[scale=1]
 \tkzDefPoints{0/0/O,2/2/A,4/0/B,6/2/C}
 \tkzDrawSegments(O,A A,B)
 \tkzDrawPoints(O,A,B)
 \tkzDrawLine(O,B)
 \tkzMarkSegments[mark=||,size=6pt](O,A A,B)
\end{tikzpicture}
```

30.11. Another marking




```
\begin{tikzpicture}[scale=1]
 \tkzDefPoint(0,0){A}\tkzDefPoint(3,2){B}
 \tkzDefPoint(4,0){C}\tkzDefPoint(2.5,1){P}
 \tkzDrawPolygon(A,B,C)
 \tkzDefEquilateral(A,P) \tkzGetPoint{P'}
 \tkzDefPointsBy[rotation=center A angle 60](P,B){P',C'}
 \tkzDrawPolygon(A,P,P')
 \tkzDrawPolySeg(P',C',A,P,B)
 \tkzDrawSegment(C,P)
 \tkzDrawPoints(A,B,C,C',P,P')
 \tkzMarkSegments[mark=s|,size=6pt,
 color=blue](A,P P,P' P',A)
 \tkzMarkSegments[mark=||,color=orange](B,P P',C')
 \tkzLabelPoints(A,C) \tkzLabelPoints[below](P)
 \tkzLabelPoints[above right](P',C',B)
\end{tikzpicture}
```

30.12. Mark an arc \tkzMarkArc

`\tkzMarkArc[⟨local options⟩](⟨pt1,pt2,pt3⟩)`

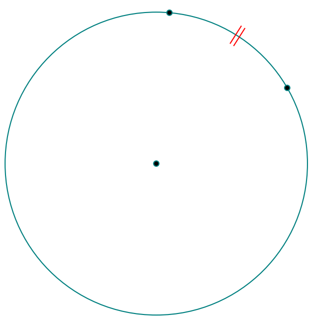
The macro allows you to place a mark on an arc. pt1 is the center, pt2 and pt3 are the endpoints of the arc.

options	default	definition
pos	.5	position of the mark
color	black	color of the mark
mark	none	choice of the mark
size	4pt	size of the mark

Possible marks are those provided by Ti*k*Z, but other marks have been created based on an idea by Yves Combe.

`|, ||,|||, z, s, x, o, oo`

30.12.1. Several marks



```
\begin{tikzpicture}
\tkzDefPoint(0,0){O}
\pgfmathsetmacro\r{2}
\tkzDefPoint(30:\r){A}
\tkzDefPoint(85:\r){B}
\tkzDrawCircle(O,A)
\tkzMarkArc[color=red,mark=||](O,A,B)
\tkzDrawPoints(B,A,O)
\end{tikzpicture}
```

30.13. Mark an angle mark : \tkzMarkAngle

More delicate operation because there are many options. The symbols used for marking in addition to those of Ti*k*Z are defined in the file `tkz-lib-marks.tex` and designated by the following characters:

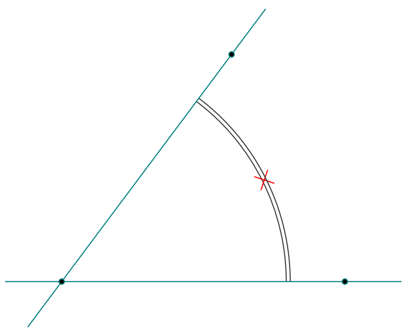
|, ||,|||, z, s, x, o, oo

\tkzMarkAngle[⟨local options⟩](⟨A,O,B⟩)

O is the vertex. Attention the arguments vary according to the options. Several markings are possible. You can simply draw an arc or add a mark on this arc. The style of the arc is chosen with the option **arc**, the radius of the arc is given by **mksize**, the arc can, of course, be colored.

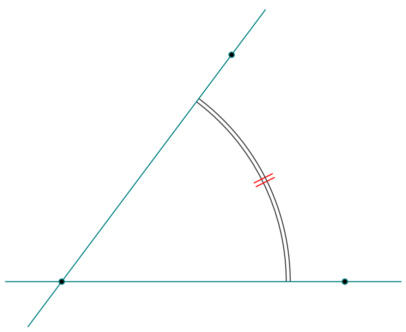
options	default	definition
arc	l	choice of l, ll and lll (single, double or triple).
size	1 (cm)	arc radius.
mark	none	choice of mark.
mksize	4pt	symbol size (mark).
mkcolor	black	symbol color (mark).
mkpos	0.5	position of the symbol on the arc.

30.13.1. Example with mark = x



```
\begin{tikzpicture}[scale=.75]
    \tkzDefPoints{0/0/O,5/0/A,3/4/B}
    \tkzMarkAngle[size = 4,mark = x,
                  arc=ll,mkcolor = red](A,O,B)
    \tkzDrawLines(O,A O,B)
    \tkzDrawPoints(O,A,B)
\end{tikzpicture}
```

30.13.2. Example with mark =||



```
\begin{tikzpicture}[scale=.75]
    \tkzDefPoints{0/0/O,5/0/A,3/4/B}
    \tkzMarkAngle[size = 4,mark = ||,
                arc=ll,mkcolor = red](A,O,B)
    \tkzDrawLines(O,A O,B)
    \tkzDrawPoints(O,A,B)
\end{tikzpicture}
```

\tkzMarkAngles[⟨local options⟩](⟨A,O,B⟩)(⟨A',O',B'⟩)etc.

With common options, there is a macro for multiple angles.

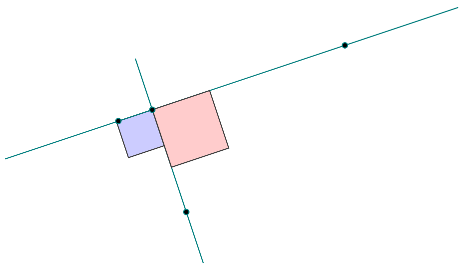
30.14. Marking a right angle: \tkzMarkRightAngle

`\tkzMarkRightAngle[⟨local options⟩](⟨A,O,B⟩)`

The `german` option allows you to change the style of the drawing. The option `size` allows to change the size of the drawing.

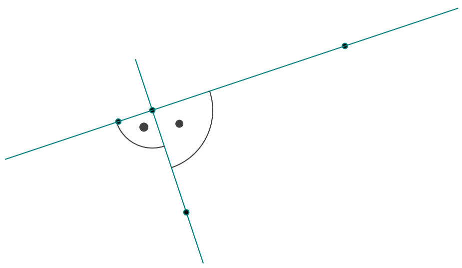
options	default	definition
german	normal	german arc with inner point.
size	0.2	side size.

30.14.1. Example of marking a right angle



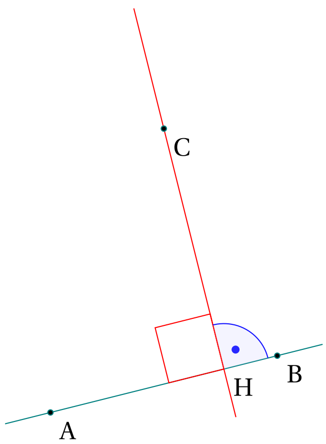
```
\begin{tikzpicture}
  \tkzDefPoints{0/0/A,3/1/B,0.9/-1.2/P}
  \tkzDefPointBy[projection = onto B--A](P)  \tkzGetPoint{H}
  \tkzDrawLines[add=.5 and .5](P,H)
  \tkzMarkRightAngle[fill=blue!20,size=.5,draw](A,H,P)
  \tkzDrawLines[add=.5 and .5](A,B)
  \tkzMarkRightAngle[fill=red!20,size=.8](B,H,P)
  \tkzDrawPoints[](A,B,P,H)
\end{tikzpicture}
```

30.14.2. Example of marking a right angle, german style



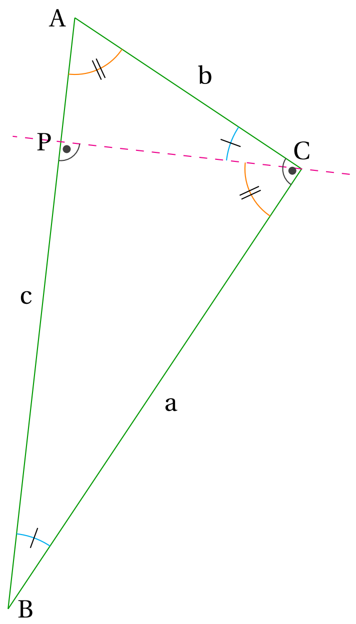
```
\begin{tikzpicture}
  \tkzDefPoints{0/0/A,3/1/B,0.9/-1.2/P}
  \tkzDefPointBy[projection = onto B--A](P)  \tkzGetPoint{H}
  \tkzDrawLines[add=.5 and .5](P,H)
  \tkzMarkRightAngle[german,size=.5,draw](A,H,P)
  \tkzDrawPoints[](A,B,P,H)
  \tkzDrawLines[add=.5 and .5](A,B)
  \tkzMarkRightAngle[german,size=.8](P,H,B)
\end{tikzpicture}
```

30.14.3. Mix of styles




```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(4,1){B}
  \tkzDefPoint(2,5){C}
  \tkzDefPointBy[projection=onto B--A](C)
      \tkzGetPoint{H}
  \tkzDrawLine(A,B)
  \tkzDrawLine[add = .5 and .2,color=red](C,H)
  \tkzMarkRightAngle[,size=1,color=red](C,H,A)
  \tkzMarkRightAngle[german,size=.8,color=blue](B,H,C)
  \tkzFillAngle[opacity=.2,fill=blue!20,size=.8](B,H,C)
  \tkzLabelPoints(A,B,C,H)
  \tkzDrawPoints(A,B,C)
\end{tikzpicture}
```

30.14.4. Full example




```
\begin{tikzpicture}[rotate=-90]
\tkzDefPoint(0,1){A}
\tkzDefPoint(2,4){C}
\tkzDefPointWith[orthogonal normed,K=7](C,A)
\tkzGetPoint{B}
\tkzDrawSegment[green!60!black](A,C)
\tkzDrawSegment[green!60!black](C,B)
\tkzDrawSegment[green!60!black](B,A)
\tkzDefSpcTriangle[orthic](A,B,C){N,O,P}
\tkzDrawLine[dashed,color=magenta](C,P)
\tkzLabelPoint[left](A){$A$}
\tkzLabelPoint[right](B){$B$}
\tkzLabelPoint[above](C){$C$}
\tkzLabelPoint[left](P){$P$}
\tkzLabelSegment[auto](B,A){$c$}
\tkzLabelSegment[auto,swap](B,C){$a$}
\tkzLabelSegment[auto,swap](C,A){$b$}
\tkzMarkAngle[size=1,color=cyan,mark=|](C,B,A)
\tkzMarkAngle[size=1,color=cyan,mark=|](A,C,P)
\tkzMarkAngle[size=0.75,color=orange,
    mark=||](P,C,B)
\tkzMarkAngle[size=0.75,color=orange,
   mark=||](B,A,C)
\tkzMarkRightAngle[german](A,C,B)
\tkzMarkRightAngle[german](B,P,C)
\end{tikzpicture}
```

30.15. \tkzMarkRightAngles

`\tkzMarkRightAngles[⟨local options⟩](⟨A,O,B⟩)(⟨A',O',B'⟩)etc.`

With common options, there is a macro for multiple angles.

Part VI.

Labelling

31. Labelling

31.1. Label for a point

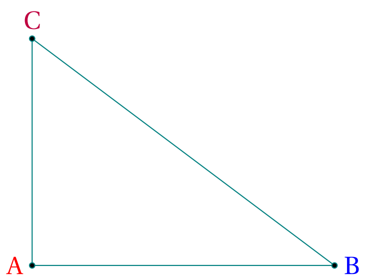
It is possible to add several labels at the same point by using this macro several times.

`\tkzLabelPoint[⟨local options⟩](⟨point⟩){⟨label⟩}`

arguments	example	
point	`\tkzLabelPoint(A){$A_1$}`	
options	default	definition
TikZ options		colour, position etc.

Optionally, we can use any style of TikZ, especially placement with above, right, dots...

31.1.1. Example with \tkzLabelPoint




```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(4,0){B}
  \tkzDefPoint(0,3){C}
  \tkzDrawSegments(A,B B,C C,A)
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoint[left,red](A){$A$}
  \tkzLabelPoint[right,blue](B){$B$}
  \tkzLabelPoint[above,purple](C){$C$}
\end{tikzpicture}
```

31.1.2. Label and reference

The reference of a point is the object that allows to use the point, the label is the name of the point that will be displayed.


```
\begin{tikzpicture}
   \tkzDefPoint(2,0){A}
   \tkzDrawPoint(A)
   \tkzLabelPoint[above](A){$A_1$}
 \end{tikzpicture}
```

31.2. Add labels to points \tkzLabelPoints

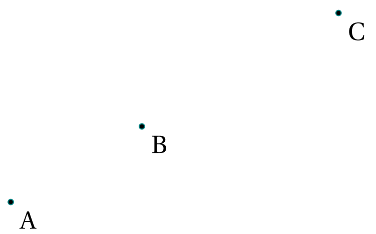
It is possible to place several labels quickly when the point references are identical to the labels and when the labels are placed in the same way in relation to the points. By default, **below right** is chosen.

`\tkzLabelPoints[⟨local options⟩](⟨`$A_1, A_2, \ldots$`⟩)`

arguments	example	result
list of points	`\tkzLabelPoints(A,B,C)`	Display of A, B and C

This macro reduces the number of lines of code, but it is not obvious that all points need the same label positioning.

31.2.1. Example with \tkzLabelPoints




```
\begin{tikzpicture}
  \tkzDefPoint(2,3){A}
  \tkzDefShiftPoint[A](30:2){B}
  \tkzDefShiftPoint[A](30:5){C}
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(A,B,C)
\end{tikzpicture}
```

31.3. Automatic position of labels \tkzAutoLabelPoints

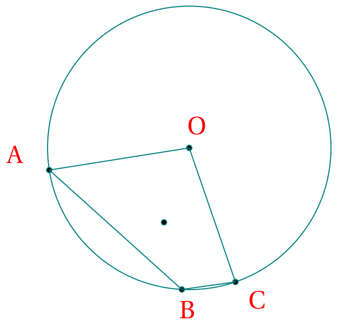
The label of a point is placed in a direction defined by a center and a point **center**. The distance to the point is determined by a percentage of the distance between the center and the point. This percentage is given by **dist**.

\tkzLabelPoints[⟨local options⟩](⟨A_1,A_2,...⟩)

arguments	example	result
list of points	\tkzLabelPoint(A,B,C)	Display of A, B and C

31.3.1. Example with \tkzAutoLabelPoints

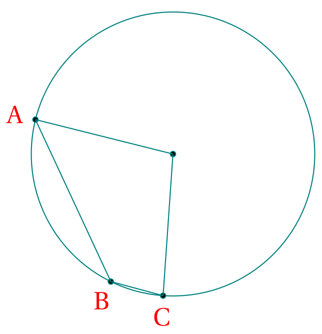
Here the points are positioned relative to the center of gravity of A, B, C and O.




```
\begin{tikzpicture}[scale=1.25]
  \tkzDefPoint(2,1){O}
  \tkzDefRandPointOn[circle=center O radius 1.5]
  \tkzGetPoint{A}
  \tkzDrawCircle(O,A)
  \tkzDefPointBy[rotation=center O angle 100](A)
  \tkzGetPoint{C}
  \tkzDefPointBy[rotation=center O angle 78](A)
  \tkzGetPoint{B}
  \tkzDrawPoints(O,A,B,C)
  \tkzDrawSegments(C,B B,A A,O O,C)
  \tkzDefCentroid(A,B,C,O)
  \tkzDrawPoint(tkzPointResult)
  \tkzAutoLabelPoints[center=tkzPointResult,
                     dist=.3,red](O,A,B,C)
\end{tikzpicture}
```

31.3.2. Example with \tkzAutoLabelPoints

This time the reference is O and the distance is by default 0.15.




```
\begin{tikzpicture}[scale=1.25]
 \tkzDefPoint(2,1){O}
 \tkzDefRandPointOn[circle=center O radius 1.5]
 \tkzGetPoint{A}
 \tkzDrawCircle(O,A)
 \tkzDefPointBy[rotation=center O angle 100](A)
 \tkzGetPoint{C}
 \tkzDefPointBy[rotation=center O angle 78](A)
 \tkzGetPoint{B}
 \tkzDrawPoints(O,A,B,C)
 \tkzDrawSegments(C,B B,A A,O O,C)
 \tkzAutoLabelPoints[center=O,red](A,B,C)
\end{tikzpicture}
```

32. Label for a segment

\tkzLabelSegment[⟨local options⟩](⟨pt1,pt2⟩){⟨label⟩}

This macro allows you to place a label along a segment or a line. The options are those of Ti*k*Z for example **pos**.

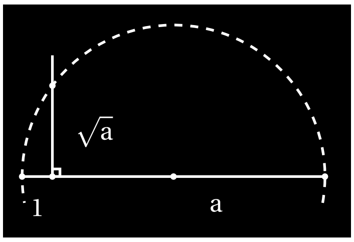
argument	example	definition
label	**\tkzLabelSegment(A,B){5}**	label text
(pt1,pt2)	(A,B)	label along [AB]

options	default	definition
pos	.5	label's position

32.0.1. First example

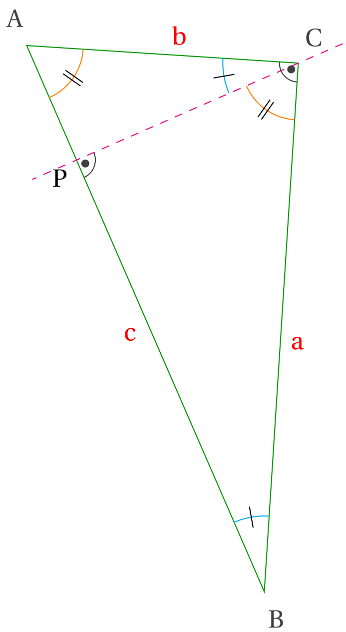

```
\begin{tikzpicture}
\tkzDefPoint(0,0){A}
\tkzDefPoint(6,0){B}
\tkzDrawSegment(A,B)
\tkzLabelSegment[above,pos=.8](A,B){$a$}
\tkzLabelSegment[below,pos=.2](A,B){$4$}
\end{tikzpicture}
```

32.0.2. Example : blackboard




```
\tikzstyle{background rectangle}=[fill=black]
\begin{tikzpicture}[show background rectangle,scale=.4]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(1,0){I}
  \tkzDefPoint(10,0){A}
  \tkzDefPointWith[orthogonal normed,K=4](I,A)
   \tkzGetPoint{H}
  \tkzDefMidPoint(O,A) \tkzGetPoint{M}
  \tkzInterLC(I,H)(M,A)\tkzGetPoints{C}{B}
  \tkzDrawSegments[color=white,line width=1pt](I,H O,A)
  \tkzDrawPoints[color=white](O,I,A,B,M)
  \tkzMarkRightAngle[color=white,line width=1pt](A,I,B)
  \tkzDrawArc[color=white,line width=1pt,
            style=dashed](M,A)(O)
  \tkzLabelSegment[white,right=1ex,pos=.5](I,B){$\sqrt{a}$}
  \tkzLabelSegment[white,below=1ex,pos=.5](O,I){$1$}
  \tkzLabelSegment[pos=.6,white,below=1ex](I,A){$a$}
\end{tikzpicture}
```

32.0.3. Labels and option : swap

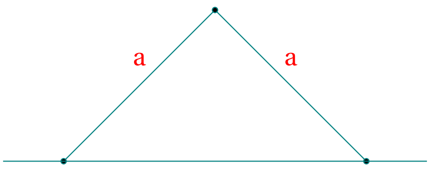



```
\begin{tikzpicture}[rotate=-60]
\tkzSetUpStyle[red,auto]{label seg style}
\tkzDefPoint(0,1){A}
\tkzDefPoint(2,4){C}
\tkzDefPointWith[orthogonal normed,K=7](C,A)
\tkzGetPoint{B}
\tkzDefSpcTriangle[orthic](A,B,C){N,O,P}
\tkzDefTriangleCenter[circum](A,B,C)
\tkzGetPoint{O}
\tkzDrawPolygon[green!60!black](A,B,C)
\tkzDrawLine[dashed,color=magenta](C,P)
\tkzLabelSegment(B,A){$c$}
\tkzLabelSegment[swap](B,C){$a$}
\tkzLabelSegment[swap](C,A){$b$}
\tkzMarkAngles[size=1,
     color=cyan,mark=|](C,B,A A,C,P)
\tkzMarkAngle[size=0.75,
     color=orange,mark=||](P,C,B)
\tkzMarkAngle[size=0.75,
      color=orange,mark=||](B,A,C)
\tkzMarkRightAngles[german](A,C,B B,P,C)
\tkzAutoLabelPoints[center = O,dist= .1](A,B,C)
 \tkzLabelPoint[below left](P){$P$}
 \end{tikzpicture}
```

\tkzLabelSegments[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩)

The arguments are a two-point couple list. The styles of Ti*k*Z are available for plotting.

32.0.4. Labels for an isosceles triangle




```
\begin{tikzpicture}[scale=1]
 \tkzDefPoints{0/0/O,2/2/A,4/0/B,6/2/C}
 \tkzDrawSegments(O,A A,B)
 \tkzDrawPoints(O,A,B)
 \tkzDrawLine(O,B)
 \tkzLabelSegments[color=red,above=4pt](O,A A,B){$a$}
\end{tikzpicture}
```

33. Add labels on a straight line \tkzLabelLine

`\tkzLabelLine[⟨local options⟩](⟨pt1,pt2⟩){⟨label⟩}`

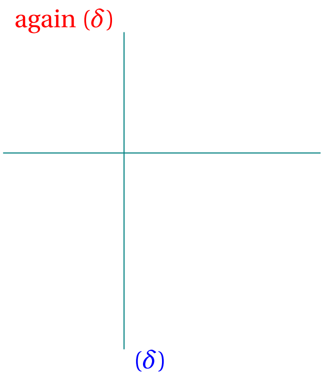
arguments	default	definition
label		`\tkzLabelLine(A,B){$\Delta$}`

options	default	definition
pos	.5	`pos` is an option for TikZ, but essential in this case…

As an option, and in addition to the `pos`, you can use all styles of Ti*k*Z, especially the placement with **above**, **right**, …

33.0.1. Example with \tkzLabelLine

An important option is **pos**, it's the one that allows you to place the label along the right. The value of **pos** can be greater than 1 or negative.




```
\begin{tikzpicture}
   \tkzDefPoints{0/0/A,3/0/B,1/1/C}
   \tkzDefLine[perpendicular=through C,K=-1](A,B)
   \tkzGetPoint{c}
   \tkzDrawLines(A,B C,c)
   \tkzLabelLine[pos=1.25,blue,right](C,c){$(\delta)$}
   \tkzLabelLine[pos=-0.25,red,left](C,c){again $(\delta)$}
\end{tikzpicture}
```

33.1. Label at an angle : \tkzLabelAngle

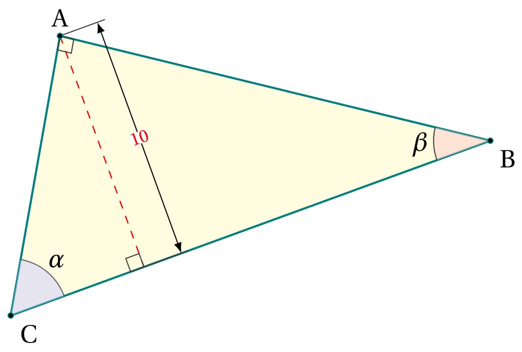
`\tkzLabelAngle[⟨local options⟩](⟨A,O,B⟩)`

There is only one option, dist (with or without unit), which can be replaced by the TikZ's pos option (without unit for the latter). By default, the value is in centimeters.

options	default	definition
pos	1	or dist, controls the distance from the top to the label.

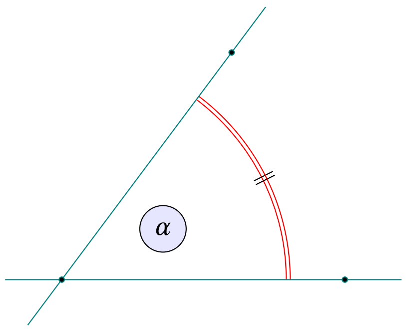
It is possible to move the label with all TikZ options : rotate, shift, below, etc.

33.1.1. Example author js bibra stackexchange

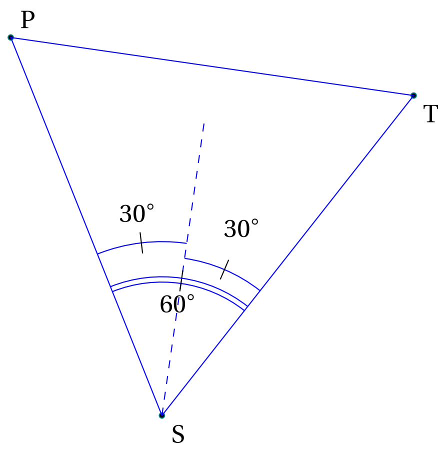



```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoint(0,0){C}
  \tkzDefPoint(20:9){B}
  \tkzDefPoint(80:5){A}
  \tkzDefPointsBy[projection=onto B--C](A){a}
  \tkzDrawPolygon[thick,fill=yellow!15](A,B,C)
  \tkzDrawSegment[dashed, red](A,a)
  \tkzDrawSegment[style=red, dashed,
  dim={$10$,15pt,midway,font=\scriptsize,
   rotate=90}](A,a)
  \tkzMarkAngle(B,C,A)
  \tkzMarkRightAngle(A,a,C)
  \tkzMarkRightAngle(C,A,B)
  \tkzFillAngle[fill=blue!20, opacity=0.5](B,C,A)
  \tkzFillAngle[fill=red!20, opacity=0.5](A,B,C)
  \tkzLabelAngle[pos=1.25](A,B,C){$\beta$}
  \tkzLabelAngle[pos=1.25](B,C,A){$\alpha$}
  \tkzMarkAngle(A,B,C)
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(B,C)
  \tkzLabelPoints[above](A)
\end{tikzpicture}
```

33.1.2. Example with pos




```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoints{0/0/O,5/0/A,3/4/B}
  \tkzMarkAngle[size = 4,mark = ||,
      arc=ll,color = red](A,O,B)%
  \tkzDrawLines(O,A O,B)
  \tkzDrawPoints(O,A,B)
  \tkzLabelAngle[pos=2,draw,circle,
      fill=blue!10](A,O,B){$\alpha$}
\end{tikzpicture}
```




```
\begin{tikzpicture}[rotate=30]
  \tkzDefPoint(2,1){S}
  \tkzDefPoint(7,3){T}
  \tkzDefPointBy[rotation=center S angle 60](T)
  \tkzGetPoint{P}
  \tkzDefLine[bisector,normed](T,S,P)
  \tkzGetPoint{s}
  \tkzDrawPoints(S,T,P)
  \tkzDrawPolygon[color=blue](S,T,P)
  \tkzDrawLine[dashed,color=blue,add=0 and 3](S,s)
  \tkzLabelPoint[above right](P){$P$}
  \tkzLabelPoints(S,T)
  \tkzMarkAngle[size = 1.8,mark = |,arc=ll,
                  color = blue](T,S,P)
  \tkzMarkAngle[size = 2.1,mark = |,arc=l,
                  color = blue](T,S,s)
  \tkzMarkAngle[size = 2.3,mark = |,arc=l,
                  color = blue](s,S,P)
 \tkzLabelAngle[pos = 1.5](T,S,P){$60^{\circ}$}%
 \tkzLabelAngles[pos = 2.7](T,S,s s,S,P){%
                          $30^{\circ}$}%
\end{tikzpicture}
```

`\tkzLabelAngles[⟨local options⟩](⟨A,O,B⟩)(⟨A',O',B'⟩)etc.`

With common options, there is a macro for multiple angles.

It finally remains to be able to give a label to designate a circle and if several possibilities are offered, we will see here `\tkzLabelCircle`.

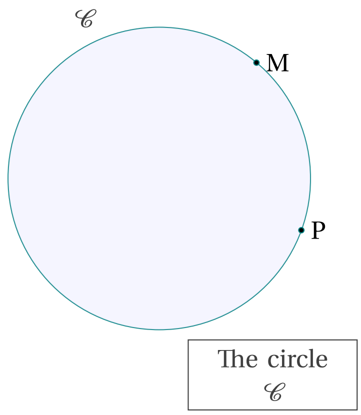
33.2. Giving a label to a circle

`\tkzLabelCircle[⟨local options⟩](⟨A,B⟩)(⟨angle⟩){⟨label⟩}`

options	default	definition
radius	radius	circle characterized by two points defining a radius
R	radius	circle characterized by a point and the measurement of a radius

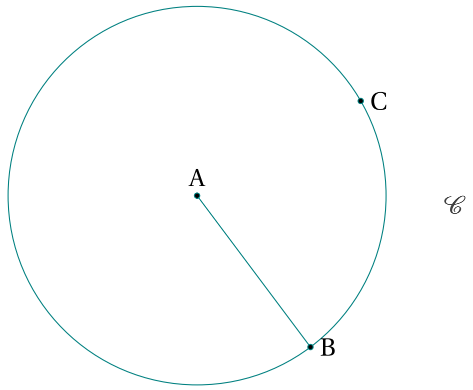
You don't need to put **radius** because that's the default option. We can use the styles from Ti*k*Z. The label is created and therefore "passed" between braces.

33.2.1. Example




```
\begin{tikzpicture}
 \tkzDefPoint(0,0){O} \tkzDefPoint(2,0){N}
 \tkzDefPointBy[rotation=center O angle 50](N)
     \tkzGetPoint{M}
 \tkzDefPointBy[rotation=center O angle -20](N)
      \tkzGetPoint{P}
 \tkzDefPointBy[rotation=center O angle 125](N)
      \tkzGetPoint{P'}
 \tkzLabelCircle[above=4pt](O,N)(120){$\mathcal{C}$}
 \tkzDrawCircle(O,M)
 \tkzFillCircle[color=blue!10,opacity=.4](O,M)
 \tkzLabelCircle[R,draw,
       text width=2cm,text centered](O,3)(-60)%
          {The circle\\ $\mathcal{C}$}
 \tkzDrawPoints(M,P)\tkzLabelPoints[right](M,P)
\end{tikzpicture}
```

33.2.2. Second example




```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoints{2/3/A,5/-1/B}
 \tkzDefPoint[label=below:$\mathcal{C}$,
              shift={(2,3)}](-30:5.5){E}
 \begin{scope}[shift=(A)]
    \tkzDefPoint(30:5){C}
 \end{scope}
 \tkzDrawCircle(A,B)
 \tkzDrawSegment(A,B)
 \tkzDrawPoints(A,B,C)
 \tkzLabelPoints[right](B,C)
 \tkzLabelPoints[above](A)
\end{tikzpicture}
```

34. Label for an arc

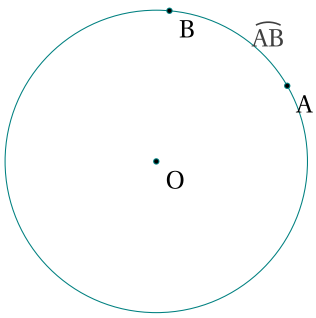
\tkzLabelArc[⟨local options⟩](⟨pt1,pt2,pt3⟩){⟨label⟩}

This macro allows you to place a label along an arc. The options are those of TikZ for example `pos`.

argument	example	definition
label	`\tkzLabelSegment(A,B){5}`	label text
(pt1,pt2,pt3)	(O,A,B)	label along the arc $\widehat{AB}$

options	default	definition
pos	.5	label's position

34.0.1. Label on arc




```
\begin{tikzpicture}
\tkzDefPoint(0,0){O}
\pgfmathsetmacro\r{2}
\tkzDefPoint(30:\r){A}
\tkzDefPoint(85:\r){B}
\tkzDrawCircle(O,A)
\tkzDrawPoints(B,A,O)
\tkzLabelArc[right=2pt](O,A,B){$\widearc{AB}$}
\tkzLabelPoints(A,B,O)
\end{tikzpicture}
```

Part VII.

Complements

35. Using the compass

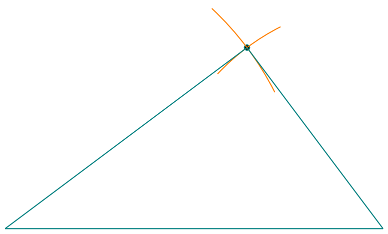
35.1. Main macro \tkzCompass

`\tkzCompass[⟨local options⟩](⟨A,B⟩)`

This macro allows you to leave a compass trace, i.e. an arc at a designated point. The center must be indicated. Several specific options will modify the appearance of the arc as well as TikZ options such as style, color, line thickness etc.
You can define the length of the arc with the option `length` or the option `delta`.

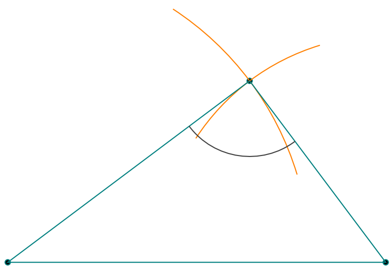
options	default	definition
delta	0 (deg)	Modifies the angle of the arc by increasing it symmetrically (in degrees)
length	1 (cm)	Changes the length (in cm)

35.1.1. Option length



```
\begin{tikzpicture}
  \tkzDefPoint(1,1){A}
  \tkzDefPoint(6,1){B}
  \tkzInterCC[R](A,4)(B,3)
  \tkzGetPoints{C}{D}
  \tkzDrawPoint(C)
  \tkzCompass[length=1.5](A,C)
  \tkzCompass(B,C)
  \tkzDrawSegments(A,B A,C B,C)
\end{tikzpicture}
```

35.1.2. Option delta



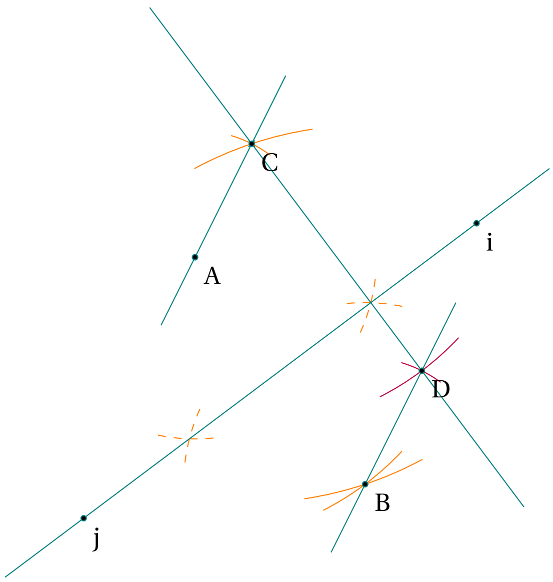
```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(5,0){B}
  \tkzInterCC[R](A,4)(B,3)
  \tkzGetPoints{C}{D}
  \tkzDrawPoints(A,B,C)
  \tkzCompass[delta=20](A,C)
  \tkzCompass[delta=20](B,C)
  \tkzDrawPolygon(A,B,C)
  \tkzMarkAngle(A,C,B)
\end{tikzpicture}
```

35.2. Multiple constructions \tkzCompasss

`\tkzCompasss[⟨local options⟩](⟨pt1,pt2 pt3,pt4,...⟩)`

Attention the arguments are lists of two points. This saves a few lines of code.

options	default	definition
delta	0	Modifies the angle of the arc by increasing it symmetrically
length	1	Changes the length




```
\begin{tikzpicture}[scale=.75]
 \tkzDefPoint(2,2){A}  \tkzDefPoint(5,-2){B}
 \tkzDefPoint(3,4){C}  \tkzDrawPoints(A,B)
 \tkzDrawPoint[shape=cross out](C)
 \tkzCompasss[new](A,B A,C B,C C,B)
 \tkzShowLine[mediator,new,dashed,length = 2](A,B)
 \tkzShowLine[parallel = through C,
                    color=purple,length=2](A,B)
 \tkzDefLine[mediator](A,B)
  \tkzGetPoints{i}{j}
 \tkzDefLine[parallel=through C](A,B)
   \tkzGetPoint{D}
 \tkzDrawLines[add=.6 and .6](C,D A,C B,D)
 \tkzDrawLines(i,j) \tkzDrawPoints(A,B,C,i,j,D)
 \tkzLabelPoints(A,B,C,i,j,D)
\end{tikzpicture}
```

36. The Show

36.1. Show the constructions of some lines \tkzShowLine

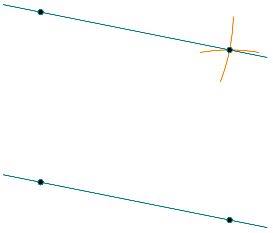
\tkzShowLine[⟨local options⟩](⟨pt1,pt2⟩) or (⟨pt1,pt2,pt3⟩)

These constructions concern mediatrices, perpendicular or parallel lines passing through a given point and bisectors. The arguments are therefore lists of two or three points. Several options allow the adjustment of the constructions. The idea of this macro comes from **Yves Combe**.

options	default	definition
mediator	mediator	displays the constructions of a mediator
perpendicular	mediator	constructions for a perpendicular
orthogonal	mediator	idem
bisector	mediator	constructions for a bisector
K	1	circle within a triangle
length	1	in cm, length of a arc
ratio	.5	arc length ratio
gap	2	placing the point of construction
size	1	radius of an arc (see bisector)

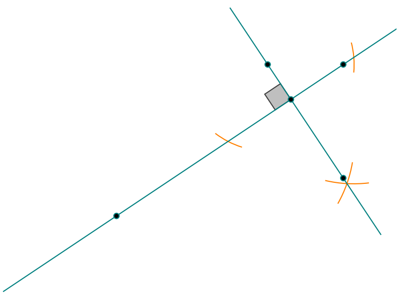
You have to add, of course, all the styles of Ti*k*Z for tracings...

36.1.1. Example of \tkzShowLine and parallel



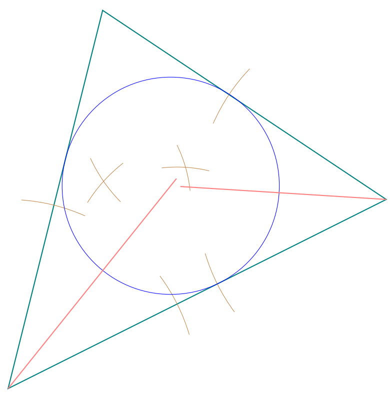
```
\begin{tikzpicture}
 \tkzDefPoints{-1.5/-0.25/A,1/-0.75/B,-1.5/2/C}
 \tkzDrawLine(A,B)
 \tkzDefLine[parallel=through C](A,B) \tkzGetPoint{c}
 \tkzShowLine[parallel=through C](A,B)
 \tkzDrawLine(C,c) \tkzDrawPoints(A,B,C,c)
\end{tikzpicture}
```

36.1.2. Example of \tkzShowLine and perpendicular



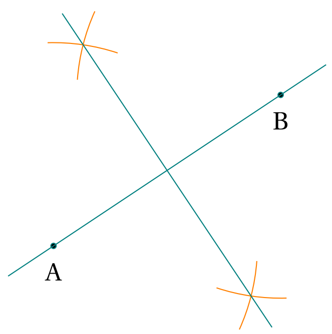
```
\begin{tikzpicture}
\tkzDefPoints{0/0/A, 3/2/B, 2/2/C}
\tkzDefLine[perpendicular=through C,K=-.5](A,B) \tkzGetPoint{c}
\tkzShowLine[perpendicular=through C,K=-.5,gap=3](A,B)
\tkzDefPointBy[projection=onto A--B](c)\tkzGetPoint{h}
\tkzMarkRightAngle[fill=lightgray](A,h,C)
\tkzDrawLines[add=.5 and .5](A,B C,c)
\tkzDrawPoints(A,B,C,h,c)
\end{tikzpicture}
```

36.1.3. Example of \tkzShowLine and bisector



```
\begin{tikzpicture}[scale=1.25]
 \tkzDefPoints{0/0/A, 4/2/B, 1/4/C}
 \tkzDrawPolygon(A,B,C)
 \tkzSetUpCompass[color=brown,line width=.1 pt]
 \tkzDefLine[bisector](B,A,C)  \tkzGetPoint{a}
 \tkzDefLine[bisector](C,B,A)  \tkzGetPoint{b}
 \tkzInterLL(A,a)(B,b) \tkzGetPoint{I}
 \tkzDefPointBy[projection = onto A--B](I)
   \tkzGetPoint{H}
 \tkzShowLine[bisector,size=2,gap=3,blue](B,A,C)
 \tkzShowLine[bisector,size=2,gap=3,blue](C,B,A)
 \tkzDrawCircle[radius,color=blue,%
 line width=.2pt](I,H)
 \tkzDrawSegments[color=red!50](I,tkzPointResult)
 \tkzDrawLines[add=0 and -0.3,color=red!50](A,a B,b)
\end{tikzpicture}
```

36.1.4. Example of \tkzShowLine and mediator




```
\begin{tikzpicture}
\tkzDefPoint(2,2){A}
\tkzDefPoint(5,4){B}
\tkzDrawPoints(A,B)
\tkzShowLine[mediator,color=orange,length=1](A,B)
\tkzGetPoints{i}{j}
\tkzDrawLines[add=-0.1 and -0.1](i,j)
\tkzDrawLines(A,B)
\tkzLabelPoints[below =3pt](A,B)
\end{tikzpicture}
```

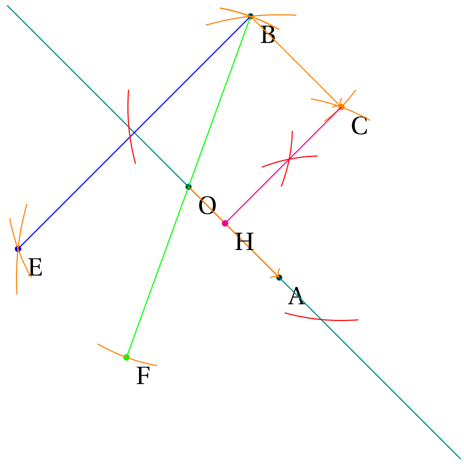
36.2. Constructions of certain transformations \tkzShowTransformation

`\tkzShowTransformation[⟨local options⟩](⟨pt1,pt2⟩) or (⟨pt1,pt2,pt3⟩)`

These constructions concern orthogonal symmetries, central symmetries, orthogonal projections and translations. Several options allow the adjustment of the constructions. The idea of this macro comes from **Yves Combe**.

options	default	definition
reflection= over pt1--pt2	reflection	constructions of orthogonal symmetry
symmetry=center pt	reflection	constructions of central symmetry
projection=onto pt1--pt2	reflection	constructions of a projection
translation=from pt1 to pt2	reflection	constructions of a translation
K	1	circle within a triangle
length	1	arc length
ratio	.5	arc length ratio
gap	2	placing the point of construction
size	1	radius of an arc (see bisector)

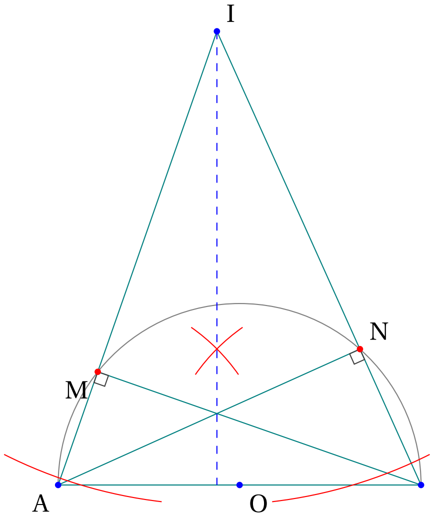
36.2.1. Example of the use of \tkzShowTransformation




```
\begin{tikzpicture}[scale=.6]
  \tkzDefPoint(0,0){O} \tkzDefPoint(2,-2){A}
  \tkzDefPoint(70:4){B} \tkzDrawPoints(A,O,B)
  \tkzLabelPoints(A,O,B)
  \tkzDrawLine[add= 2 and 2](O,A)
  \tkzDefPointBy[translation=from O to A](B)
  \tkzGetPoint{C}
  \tkzDrawPoint[color=orange](C)  \tkzLabelPoints(C)
  \tkzShowTransformation[translation=from O to A,%
           length=2](B)
  \tkzDrawSegments[->,color=orange](O,A B,C)
  \tkzDefPointBy[reflection=over O--A](B) \tkzGetPoint{E}
  \tkzDrawSegment[blue](B,E)
  \tkzDrawPoint[color=blue](E)\tkzLabelPoints(E)
  \tkzShowTransformation[reflection=over O--A,size=2](B)
  \tkzDefPointBy[symmetry=center O](B) \tkzGetPoint{F}
  \tkzDrawSegment[color=green](B,F)
  \tkzDrawPoint[color=green](F)\tkzLabelPoints(F)
  \tkzShowTransformation[symmetry=center O,%
                 length=2](B)
  \tkzDefPointBy[projection=onto O--A](C)
  \tkzGetPoint{H}
  \tkzDrawSegments[color=magenta](C,H)
  \tkzDrawPoint[color=magenta](H)\tkzLabelPoints(H)
  \tkzShowTransformation[projection=onto O--A,%
                color=red,size=3,gap=-2](C)
\end{tikzpicture}
```

36.2.2. Another example of the use of \tkzShowTransformation

You'll find this figure again, but without the construction features.




```
\begin{tikzpicture}[scale=.6]
  \tkzDefPoints{0/0/A,8/0/B,3.5/10/I}
  \tkzDefMidPoint(A,B) \tkzGetPoint{O}
  \tkzDefPointBy[projection=onto A--B](I)
     \tkzGetPoint{J}
  \tkzInterLC(I,A)(O,A)  \tkzGetPoints{M'}{M}
  \tkzInterLC(I,B)(O,A)  \tkzGetPoints{N}{N'}
  \tkzDrawSemiCircle[diameter](A,B)
  \tkzDrawSegments(I,A I,B A,B B,M A,N)
  \tkzMarkRightAngles(A,M,B A,N,B)
  \tkzDrawSegment[style=dashed,color=blue](I,J)
  \tkzShowTransformation[projection=onto A--B,
                color=red,size=3,gap=-3](I)
  \tkzDrawPoints[color=red](M,N)
  \tkzDrawPoints[color=blue](O,A,B,I)
  \tkzLabelPoints(O)
  \tkzLabelPoints[above right](N,I)
  \tkzLabelPoints[below left](M,A)
\end{tikzpicture}
```

37. Protractor

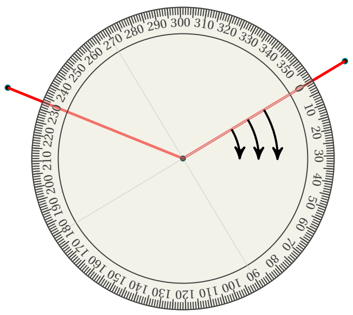
Based on an idea by Yves Combe, the following macro allows you to draw a protractor. The operating principle is even simpler. Just name a half-line (a ray). The protractor will be placed on the origin O, the direction of the half-line is given by A. The angle is measured in the direct direction of the trigonometric circle.

```
\tkzProtractor[⟨local options⟩](⟨O,A⟩)
```

options	default	definition
lw	0.4 pt	line thickness
scale	1	ratio: adjusts the size of the protractor
return	false	trigonometric circle indirect

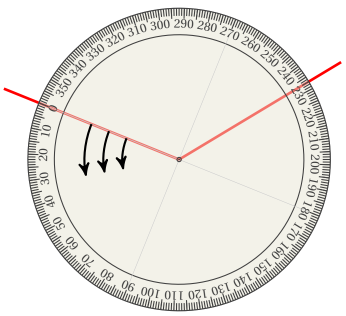
37.1. The circular protractor

Measuring in the forward direction



```
\begin{tikzpicture}[scale=.5]
\tkzDefPoint(2,0){A}\tkzDefPoint(0,0){O}
\tkzDefShiftPoint[A](31:5){B}
\tkzDefShiftPoint[A](158:5){C}
\tkzDrawPoints(A,B,C)
\tkzDrawSegments[color = red,
    line width = 1pt](A,B A,C)
  \tkzProtractor[scale = 1](A,B)
\end{tikzpicture}
```

37.2. The circular protractor, transparent and returned



```
\begin{tikzpicture}[scale=.5]
  \tkzDefPoint(2,3){A}
  \tkzDefShiftPoint[A](31:5){B}
   \tkzDefShiftPoint[A](158:5){C}
  \tkzDrawSegments[color=red,line width=1pt](A,B A,C)
  \tkzProtractor[return](A,C)
\end{tikzpicture}
```

38. Miscellaneous tools

38.1. Duplicate a segment

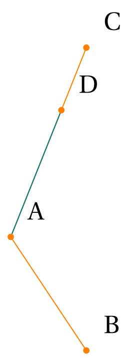
This involves constructing a segment on a given half-line of the same length as a given segment.

`\tkzDuplicateSegment(⟨pt1,pt2⟩)(⟨pt3,pt4⟩){⟨pt5⟩}`

This involves creating a segment on a given half-line of the same length as a given segment . It is in fact the definition of a point. `\tkzDuplicateSegment` is the new name of `\tkzDuplicateLen`.

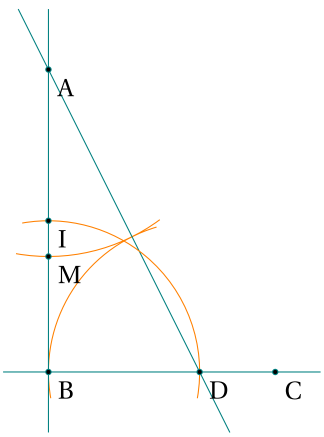
arguments	example	explication
(pt1,pt2)(pt3,pt4){pt5}	`\tkzDuplicateSegment(A,B)(E,F){C}`	AC=EF and C ∈ [AB)

The macro `\tkzDuplicateLength` is identical to this one.



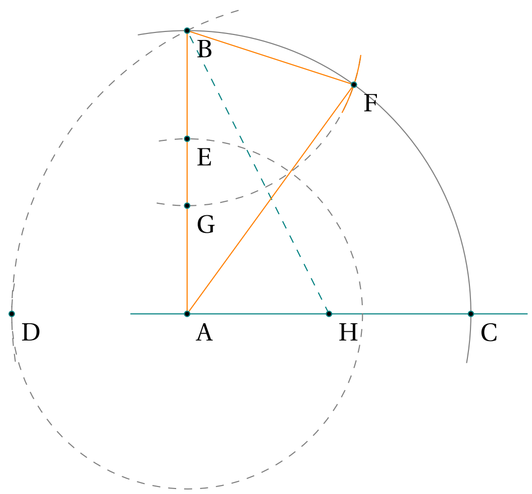

```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoints{0/0/A,2/-3/B,2/5/C}
 \tkzDuplicateSegment(A,B)(A,C)
 \tkzGetPoint{D}
 \tkzDrawSegments[new](A,B A,C)
 \tkzDrawSegment[teal](A,D)
 \tkzDrawPoints[new](A,B,C,D)
 \tkzLabelPoints[above right=3pt](A,B,C,D)
\end{tikzpicture}
```

38.1.1. Proportion of gold with \tkzDuplicateSegment




```
\begin{tikzpicture}[rotate=-90,scale=.4]
 \tkzDefPoints{0/0/A,10/0/B}
 \tkzDefMidPoint(A,B)
 \tkzGetPoint{I}
 \tkzDefPointWith[orthogonal,K=-.75](B,A)
 \tkzGetPoint{C}
 \tkzInterLC(B,C)(B,I)  \tkzGetSecondPoint{D}
 \tkzDuplicateSegment(B,D)(D,A) \tkzGetPoint{E}
 \tkzInterLC(A,B)(A,E)   \tkzGetPoints{N}{M}
 \tkzDrawArc[orange,delta=10](D,E)(B)
 \tkzDrawArc[orange,delta=10](A,M)(E)
 \tkzDrawLines(A,B B,C A,D)
 \tkzDrawArc[orange,delta=10](B,D)(I)
 \tkzDrawPoints(A,B,D,C,M,I)
 \tkzLabelPoints(A,B,D,C,M,I)
\end{tikzpicture}
```

38.1.2. Golden triangle or sublime triangle




```
\begin{tikzpicture}[scale=.75]
  \tkzDefPoints{0/0/A,5/0/C,0/5/B}
  \tkzDefMidPoint(A,C)\tkzGetPoint{H}
  \tkzDuplicateSegment(H,B)(H,A)\tkzGetPoint{D}
  \tkzDuplicateSegment(A,D)(A,B)\tkzGetPoint{E}
  \tkzDuplicateSegment(A,D)(B,A)\tkzGetPoint{G}
  \tkzInterCC(A,C)(B,G)\tkzGetSecondPoint{F}
  \tkzDrawLine(A,C)
  \tkzDrawArc(A,C)(B)
  \begin{scope}[arc style/.style={color=gray,%
                              style=dashed}]
    \tkzDrawArc(H,B)(D)
    \tkzDrawArc(A,D)(B)
    \tkzDrawArc(B,G)(F)
  \end{scope}
  \tkzDrawSegment[dashed](H,B)
  \tkzCompass(B,F)
  \tkzDrawPolygon[new](A,B,F)
  \tkzDrawPoints(A,...,H)
  \tkzLabelPoints(A,...,H)
\end{tikzpicture}
```

38.2. Segment length \tkzCalcLength

There's an option in Ti*k*Z named **veclen**. This option is used to calculate AB if A and B are two points.
The only problem for me is that the version of Ti*k*Z is not accurate enough in some cases. My version uses the `xfp` package and is slower, but more accurate.

`\tkzCalcLength[⟨local options⟩](⟨pt1,pt2⟩)`

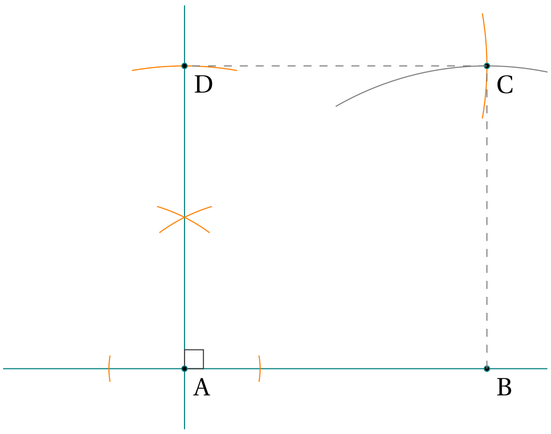
You can store the result with the macro `\tkzGetLength` for example `\tkzGetLength{dAB}` defines the macro `\dAB`.

arguments	example	explication
`(pt1,pt2){name of macro}`	`\tkzCalcLength[pt](A,B)`	`\dAB gives AB in pt`

Only one option

options	default	example
`cm`	`true`	`\tkzCalcLength(A,B) After \tkzGetLength{dAB} \dAB gives AB in cm`

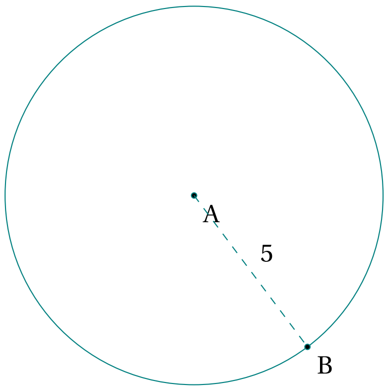
38.2.1. Compass square construction




```
\begin{tikzpicture}[scale=1]
  \tkzDefPoint(0,0){A} \tkzDefPoint(4,0){B}
  \tkzCalcLength(A,B)\tkzGetLength{dAB}
  \tkzDefLine[perpendicular=through A](A,B)
  \tkzGetPoint{D}
  \tkzDefPointWith[orthogonal,K=-1](B,A)
    \tkzGetPoint{F}
  \tkzGetPoint{C}
  \tkzDrawLine[add= .6 and .2](A,B)
  \tkzDrawLine(A,D)
  \tkzShowLine[orthogonal=through A,gap=2](A,B)
  \tkzMarkRightAngle(B,A,D)
  \tkzCompasss(A,D D,C)
  \tkzDrawArc[R](B,\dAB)(80,110)
  \tkzDrawPoints(A,B,C,D)
  \tkzDrawSegments[color=gray,style=dashed](B,C C,D)
  \tkzLabelPoints(A,B,C,D)
\end{tikzpicture}
```

38.2.2. Example

The macro `\tkzDefCircle[radius](A,B)` defines the radius that we retrieve with `\tkzGetLength`, this result is in `cm`.




```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoint(0,0){A}
 \tkzDefPoint(3,-4){B}
 \tkzDefCircle[through](A,B)
 \tkzGetLength{rABcm}
 \tkzDrawCircle(A,B)
 \tkzDrawPoints(A,B)
 \tkzLabelPoints(A,B)
 \tkzDrawSegment[dashed](A,B)
 \tkzLabelSegment(A,B){$\pgfmathprintnumber{\rABcm}$}
\end{tikzpicture}
```

38.3. Transformation from pt to cm or cm to pt

Not sure if this is necessary and it is only a division by 28.45274 and a multiplication by the same number. The macros are:

`\tkzpttocm(⟨nombre⟩){⟨name of macro⟩}`

The result is stored in a macro.

arguments	example	explication
(nombre)name of macro	`\tkzpttocm(120){len}`	`\len donne un nombre de tkznamecm`

You'll have to use `\len` along with `cm`.

38.4. Change of unit

`\tkzcmtopt(⟨nombre⟩){⟨name of macro⟩}`

The result is stored in a macro.

arguments	example	explication
(nombre){name of macro}	\tkzcmtopt(5){len}	\len longueur en pts

The result can be used with `\lenpt`

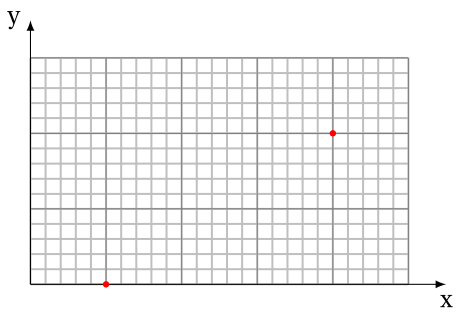
38.5. Get point coordinates

`\tkzGetPointCoord(⟨A⟩){⟨name of macro⟩}`

arguments	example	explication
(point){name of macro}	\tkzGetPointCoord(A){A}	\Ax and \Ay give coordinates for A

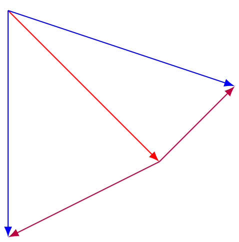
Stores in two macros the coordinates of a point. If the name of the macro is `p`, then `\px` and `\py` give the coordinates of the chosen point with the cm as unit.

38.5.1. Coordinate transfer with \tkzGetPointCoord




```
\begin{tikzpicture}
 \tkzInit[xmax=5,ymax=3]
 \tkzGrid[sub,orange]
 \tkzDrawX \tkzDrawY
 \tkzDefPoint(1,0){A}
 \tkzDefPoint(4,2){B}
 \tkzGetPointCoord(A){a}
 \tkzGetPointCoord(B){b}
 \tkzDefPoint(\ax,\ay){C}
 \tkzDefPoint(\bx,\by){D}
 \tkzDrawPoints[color=red](C,D)
\end{tikzpicture}
```

38.5.2. Sum of vectors with \tkzGetPointCoord



```
\begin{tikzpicture}[>=latex]
  \tkzDefPoint(1,4){a}
  \tkzDefPoint(3,2){b}
  \tkzDefPoint(1,1){c}
  \tkzDrawSegment[->,red](a,b)
  \tkzGetPointCoord(c){c}
  \draw[color=blue,->](a) -- ([shift=(b)]\cx,\cy) ;
  \draw[color=purple,->](b) -- ([shift=(b)]\cx,\cy) ;
  \tkzDrawSegment[->,blue](a,c)
  \tkzDrawSegment[->,purple](b,c)
\end{tikzpicture}
```

Part VIII.

Working with style

39. Predefined styles

The way to proceed will depend on your use of the package. A method that seems to me to be correct is to use as much as possible predefined styles in order to separate the content from the form. This method will be the right one if you plan to create a document (like this documentation) with many figures. We will see how to define a global style for a document. We will see how to use a style locally.
The file **tkz-euclide.cfg** contains the predefined styles of the main objects. Among these the most important are points, lines, segments, circles, arcs and compass traces. If you always use the same styles and if you create many figures then it is interesting to create your own styles . To do this you need to know what features you can modify. It will be necessary to know some notions of Ti*k*Z.
The predefined styles are global styles. They exist before the creation of the figures. It is better to avoid changing them between two figures. On the other hand these styles can be modified in a figure temporarily. There the styles are defined locally and do not influence the other figures.
For the document you are reading here is how I defined the different styles.

```
\tkzSetUpColors[background=white,text=black]
\tkzSetUpPoint[size=2,color=teal]
\tkzSetUpLine[line width=.4pt,color=teal]
\tkzSetUpCompass[color=orange, line width=.4pt,delta=10]
\tkzSetUpArc[color=gray,line width=.4pt]
\tkzSetUpStyle[orange]{new}
```

The macro **\tkzSetUpColors** allows you to set the background color as well as the text color. If you don't use it, the colors of your document will be used as well as the fonts. Let's see how to define the styles of the main objects.

40. Points style

This is how the points are defined :

```
\tikzset{point style/.style = {%
       draw         = \tkz@euc@pointcolor,
       inner sep    = 0pt,
       shape        = \tkz@euc@pointshape,
       minimum size = \tkz@euc@pointsize,
       fill         = \tkz@euc@pointcolor}}
```

It is of course possible to use **\tikzset** but you can use a macro provided by the package. You can use the macro **\tkzSetUpPoint** globally or locally,
Let's look at this possibility.

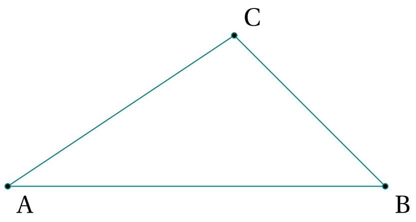
40.0.1. Use of \tkzSetUpPoint

\tkzSetUpPoint[⟨local options⟩]

options	default	definition
color	black	point color
size	3	point size
fill	black!50	inside point color
shape	circle	point shape circle, cross or cross out

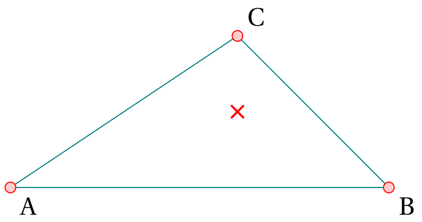
40.0.2. Global style or local style

First of all here is a figure created with the styles of my documentation, then the style of the points is modified within the environment **tikzspicture**.
You can use the macro `\tkzSetUpPoint` globally or locally, If you place this macro in your preamble or before your first figure then the point style will be valid for all figures in your document. Il sera possible d'utiliser un autre style locallement en utilisant cette commande au sein d'un environnement **tikzpicture**.
Let's look at this possibility.



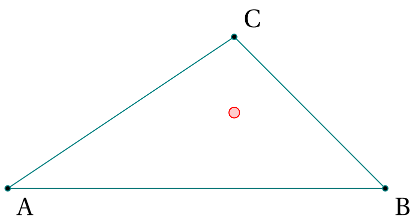

```
\begin{tikzpicture}
  \tkzDefPoints{0/0/A,5/0/B,3/2/C,3/1/D}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(A,B)
  \tkzLabelPoints[above right](C)
\end{tikzpicture}
```

The style of the points is modified locally in the second figure



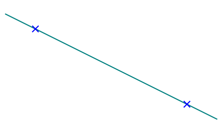

```
\begin{tikzpicture}
  \tkzSetUpPoint[size=4,color=red,fill=red!20]
  \tkzDefPoints{0/0/A,5/0/B,3/2/C,3/1/D}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawPoints(A,B,C)
  \tkzDrawPoint[shape=cross out,thick](D)
  \tkzLabelPoints(A,B)
  \tkzLabelPoints[above right](C)
\end{tikzpicture}
```

The points get back the initial style. Point D has a new style limited by the environment **scope**. It is also possible to use `{...}` orThe points get back the initial style. Point D has a new style limited by the environment **scope**. It is also possible to use `{...}` or `\begingoup ... \endgroup`.



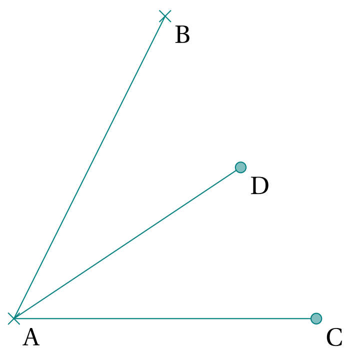

```
\begin{tikzpicture}
  \tkzDefPoints{0/0/A,5/0/B,3/2/C,3/1/D}
  \tkzDrawPolygon(A,B,C)
  \tkzDrawPoints(A,B,C)
  \begin{scope}
    \tkzSetUpPoint[size=4,color=red,fill=red!20]
    \tkzDrawPoint(D)
  \end{scope}
  \tkzLabelPoints(A,B)
  \tkzLabelPoints[above right](C)
\end{tikzpicture}
```

40.0.3. Simple example with \tkzSetUpPoint



```
\begin{tikzpicture}
  \tkzSetUpPoint[shape = cross out,color=blue]
  \tkzDefPoint(2,1){A}
  \tkzDefPoint(4,0){B}
  \tkzDrawLine(A,B)
  \tkzDrawPoints(A,B)
\end{tikzpicture}
```

40.0.4. Use of \tkzSetUpPoint inside a group




```
\begin{tikzpicture}
  \tkzDefPoints{0/0/A,2/4/B,4/0/C,3/2/D}
  \tkzDrawSegments(A,B A,C A,D)
  {\tkzSetUpPoint[shape=cross out,
            fill= teal!50,
            size=4,color=teal]
  \tkzDrawPoints(A,B)}
  \tkzSetUpPoint[fill= teal!50,size=4,
               color=teal]
   \tkzDrawPoints(C,D)
  \tkzLabelPoints(A,B,C,D)
\end{tikzpicture}
```

41. Lines style

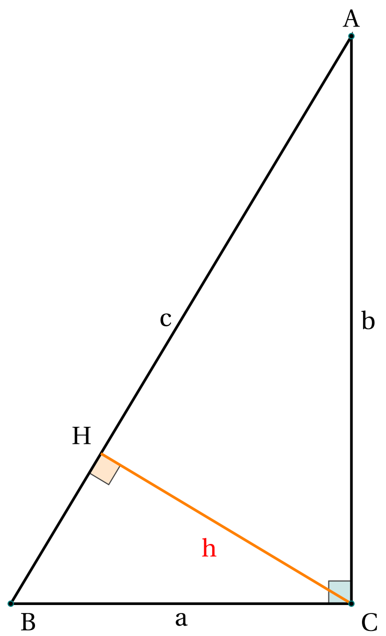
41.0.1. Use of \tkzSetUpLine

It is a macro that allows you to define the style of all the lines.

`\tkzSetUpLine[⟨local options⟩]`

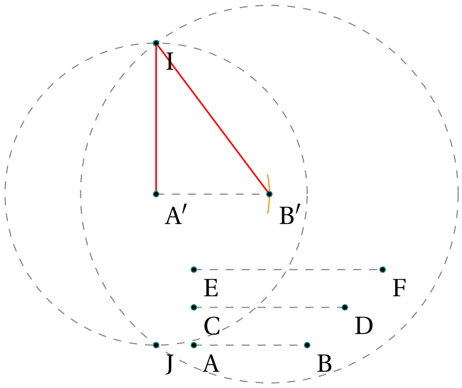
options	default	definition
color	black	colour of the construction lines
line width	0.4pt	thickness of the construction lines
style	solid	style of construction lines
add	.2 and .2	changing the length of a line segment

41.0.2. Change line width



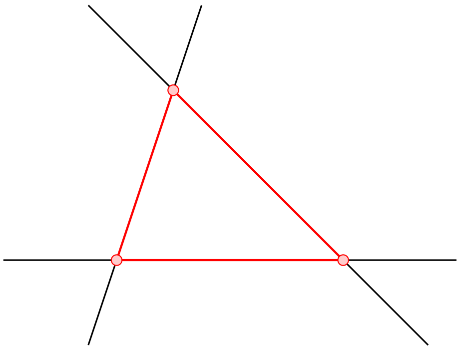

```
\begin{tikzpicture}[scale=.75]
\tkzSetUpLine[line width=1pt]
\begin{scope}[rotate=-90]
    \tkzDefPoints{0/6/A,10/0/B,10/6/C}
    \tkzDefPointBy[projection = onto B--A](C)
    \tkzGetPoint{H}
    \tkzMarkRightAngle[size=.4,
                    fill=teal!20](B,C,A)
    \tkzMarkRightAngle[size=.4,
                    fill=orange!20](B,H,C)
    \tkzDrawPolygon(A,B,C)
    \tkzDrawSegment[new](C,H)
\end{scope}
 \tkzLabelSegment[below](C,B){$a$}
 \tkzLabelSegment[right](A,C){$b$}
 \tkzLabelSegment[left](A,B){$c$}
 \tkzLabelSegment[color=red](C,H){$h$}
 \tkzDrawPoints(A,B,C)
 \tkzLabelPoints[above left](H)
 \tkzLabelPoints(B,C)
 \tkzLabelPoints[above](A)
\end{tikzpicture}
```

41.0.3. Change style of line




```
\begin{tikzpicture}[scale=.5]
\tikzset{line style/.style = {color = gray,
                              style=dashed}}
\tkzDefPoints{1/0/A,4/0/B,1/1/C,5/1/D}
\tkzDefPoints{1/2/E,6/2/F,0/4/A',3/4/B'}
\tkzCalcLength(C,D)
\tkzGetLength{rCD}
\tkzCalcLength(E,F)
\tkzGetLength{rEF}
\tkzInterCC[R](A',\rCD)(B',\rEF)
\tkzGetPoints{I}{J}
\tkzDrawLine(A',B')
\tkzCompass(A',B')
\tkzDrawSegments(A,B C,D E,F)
\tkzDrawCircles[R](A',{\rCD} B',\rEF)
\begin{scope}
  \tkzSetUpLine[color=red]
  \tkzDrawSegments(A',I B',I)
\end{scope}
\tkzDrawPoints(A,B,C,D,E,F,A',B',I,J)
\tkzLabelPoints(A,B,C,D,E,F,A',B',I,J)
\end{tikzpicture}
```

41.0.4. Example 3: extend lines

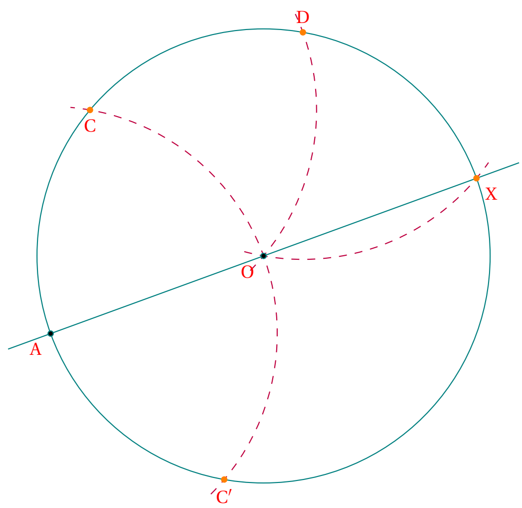


```
\begin{tikzpicture}[scale=.75]
\tkzSetUpLine[add=.5 and .5]
\tkzDefPoints{0/0/A,4/0/B,1/3/C}
\tkzDrawLines(A,B B,C A,C)
\tkzDrawPolygon[red,thick](A,B,C)
\tkzSetUpPoint[size=4,circle,color=red,fill=red!20]
\tkzDrawPoints(A,B,C)
\end{tikzpicture}
```

42. Arc style

\tkzSetUpArc[⟨local options⟩]

options	default	definition
color	black	colour of the lines
line width	0.4pt	thickness of the lines
style	solid	style of construction lines




```
\begin{tikzpicture}
\def\r{3} \def\angle{200}
\tkzSetUpArc[delta=5,color=purple,style=dashed]
\tkzSetUpLabel[font=\scriptsize,red]
\tkzDefPoint(0,0){O}
\tkzDefPoint(\angle:\r){A}
\tkzInterCC(O,A)(A,O) \tkzGetPoints{C'}{C}
\tkzInterCC(O,A)(C,O) \tkzGetPoints{D'}{D}
\tkzInterCC(O,A)(D,O) \tkzGetPoints{X'}{X}
\tkzDrawCircle(O,A)
\tkzDrawArc(A,C')(C)
\tkzDrawArc(C,O)(D)
\tkzDrawArc(D,O)(X)
\tkzDrawLine[add=.1 and .1](A,X)
\tkzDrawPoints(O,A)
\tkzDrawPoints[new](C,C',D,X)
\tkzLabelPoints[below left](O,A)
\tkzLabelPoints[below](C,C')
\tkzLabelPoints[below right](X)
\tkzLabelPoints[above](D)
\end{tikzpicture}
```

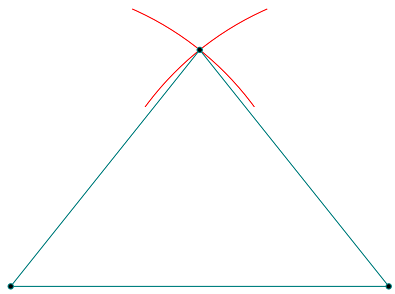
43. Compass style, configuration macro \tkzSetUpCompass

The following macro will help to understand the construction of a figure by showing the compass traces necessary to obtain certain points.

`\tkzSetUpCompass[⟨local options⟩]`

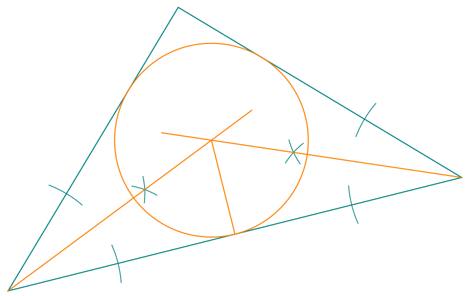
options	default	definition
color	black	colour of the construction lines
line width	0.4pt	thickness of the construction lines
style	solid	style of lines : solid, dashed,dotted,...
delta	0	changes the length of the arc

43.0.1. Use of \tkzSetUpCompass



```
\begin{tikzpicture}
  \tkzSetUpCompass[color=red,delta=15]
  \tkzDefPoint(1,1){A}
  \tkzDefPoint(6,1){B}
  \tkzInterCC[R](A,4)(B,4) \tkzGetPoints{C}{D}
  \tkzCompass(A,C)
  \tkzCompass(B,C)
  \tkzDrawPolygon(A,B,C)
  \tkzDrawPoints(A,B,C)
\end{tikzpicture}
```

43.0.2. Use of \tkzSetUpCompass with \tkzShowLine



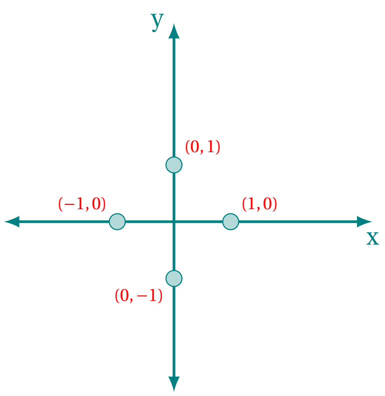
```
\begin{tikzpicture}[scale=.75]
\tkzSetUpStyle[bisector,size=2,gap=3]{showbi}
\tkzSetUpCompass[color=teal,line width=.3 pt]
\tkzDefPoints{0/1/A, 8/3/B, 3/6/C}
\tkzDrawPolygon(A,B,C)
\tkzDefLine[bisector](B,A,C) \tkzGetPoint{a}
\tkzDefLine[bisector](C,B,A) \tkzGetPoint{b}
\tkzShowLine[showbi](B,A,C)
\tkzShowLine[showbi](C,B,A)
\tkzInterLL(A,a)(B,b) \tkzGetPoint{I}
\tkzDefPointBy[projection= onto A--B](I)
\tkzGetPoint{H}
\tkzDrawCircle[radius,new](I,H)
\tkzDrawSegments[new](I,H)
\tkzDrawLines[add=0 and .2,new](A,I B,I)
\end{tikzpicture}
```

44. Label style

The macro **\tkzSetUpLabel** is used to define the style of the point labels.

\tkzSetUpStyle[⟨local options⟩]

The options are the same as those of Ti*k*Z




```
\begin{tikzpicture}[scale=.75]
  \tkzSetUpLabel[font=\scriptsize,red]
  \tkzSetUpStyle[line width=1pt,teal,<->]{XY}
  \tkzInit[xmin=-3,xmax=3,ymin=-3,ymax=3]
  \tkzDrawX[XY]
  \tkzDrawY[XY]
  \tkzDefPoints{1/0/A,0/1/B,-1/0/C,0/-1/D}
  \tkzDrawPoints[teal,fill=teal!30,size=6](A,...,D)
  \tkzLabelPoint[above right](A){$(1,0)$}
  \tkzLabelPoint[above right](B){$(0,1)$}
  \tkzLabelPoint[above left](C){$(-1,0)$}
  \tkzLabelPoint[below left](D){$(0,-1)$}
\end{tikzpicture}
```

45. Own style

You can set your own style with **\tkzSetUpStyle**

\tkzSetUpStyle[⟨local options⟩]

The options are the same as those of Ti*k*Z


```
\begin{tikzpicture}
  \tkzSetUpStyle[color=blue!20!black,fill=blue!20]{mystyle}
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(0,1){A}
  \tkzDrawPoints(O) % general style
  \tkzDrawPoints[mystyle,size=4](A) % my style
  \tkzLabelPoints(O,A)
\end{tikzpicture}
```

Part IX.

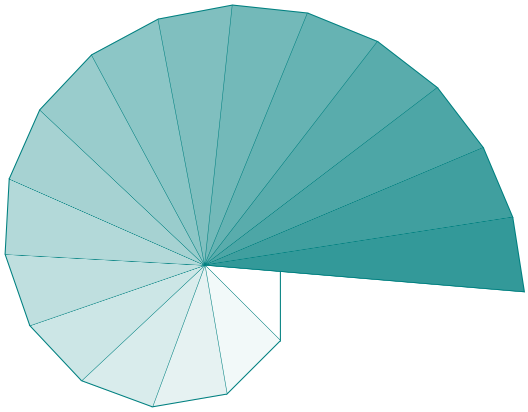
Examples

46. Some interesting examples

46.1. Square root of the integers

Square root of the integers

How to get 1, $\sqrt{2}$, $\sqrt{3}$ *with a rule and a compass.*

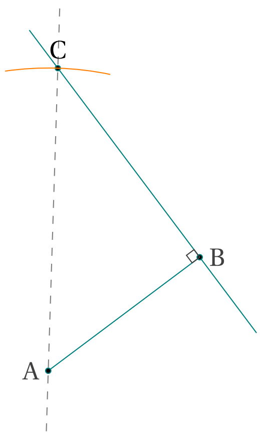


```
\begin{tikzpicture}
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(1,0){a0}
   \tkzDrawSegment(O,a0)
  \foreach \i [count=\j] in {0,...,16}{%
    \tkzDefPointWith[orthogonal normed](a\i,O)
    \tkzGetPoint{a\j}
       \pgfmathsetmacro{\c}{5*\i}
    \tkzDrawPolySeg[fill=teal!\c](a\i,a\j,O)}
 \end{tikzpicture}
```

46.2. About right triangle

About right triangle

We have a segment [AB] *and we want to determine a point* C *such that* AC = 8 *cm and* ABC *is a right triangle in* B.

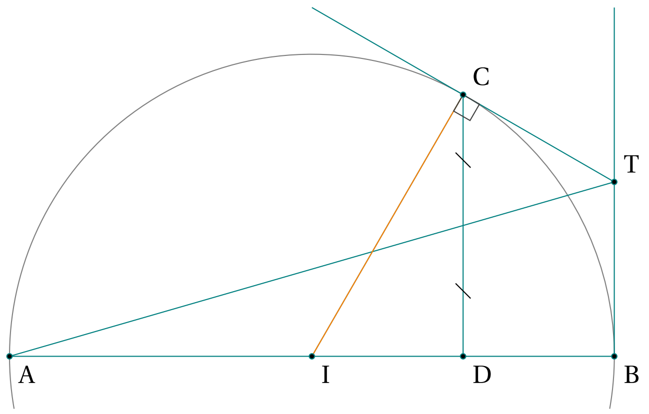



```
\begin{tikzpicture}[scale=.5]
  \tkzDefPoint["$A$" left](2,1){A}
  \tkzDefPoint["$B$" right](6,4){B}
  \tkzDefPointWith[orthogonal,K=-1](B,A)
  \tkzDrawLine[add = .5 and .5](B,tkzPointResult)
  \tkzInterLC[R](B,tkzPointResult)(A,8)
  \tkzGetPoints{C}{J}
  \tkzDrawSegment(A,B)
  \tkzDrawPoints(A,B,C)
  \tkzCompass(A,C)
  \tkzMarkRightAngle(A,B,C)
  \tkzDrawLine[color=gray,style=dashed](A,C)
  \tkzLabelPoint[above](C){$C$}
\end{tikzpicture}
```

46.3. Archimedes

Archimedes

This is an ancient problem proved by the great Greek mathematician Archimedes . The figure below shows a semicircle, with diameter AB. *A tangent line is drawn and touches the semicircle at* B. *An other tangent line at a point,* C*, on the semicircle is drawn. We project the point* C *on the line segment* [AB] *on a point* D. *The two tangent lines intersect at the point* T. *Prove that the line* (AT) *bisects* (CD)

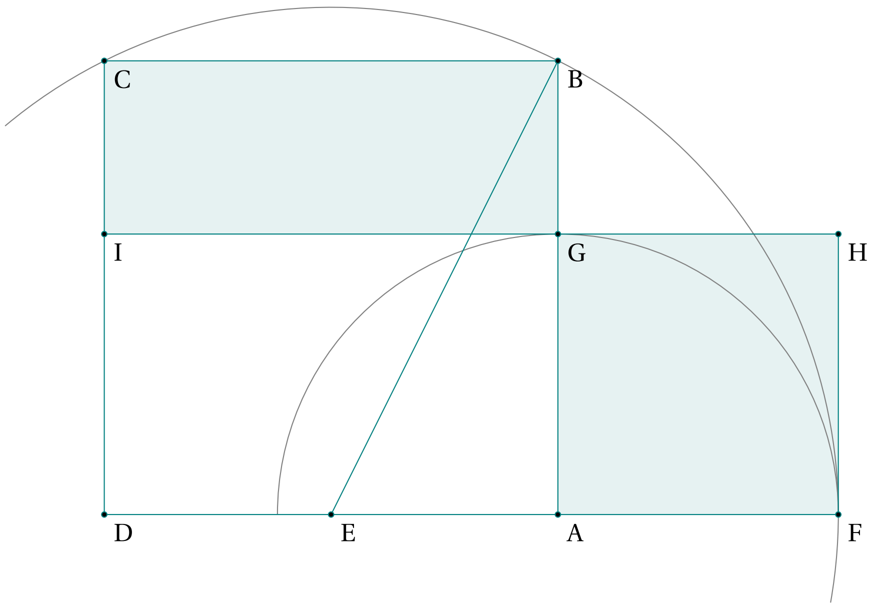



```
\begin{tikzpicture}[scale=1]
  \tkzDefPoint(0,0){A}\tkzDefPoint(6,0){D}
  \tkzDefPoint(8,0){B}\tkzDefPoint(4,0){I}
  \tkzDefLine[orthogonal=through D](A,D)
  \tkzInterLC[R](D,tkzPointResult)(I,4) \tkzGetFirstPoint{C}
  \tkzDefLine[orthogonal=through C](I,C)    \tkzGetPoint{c}
  \tkzDefLine[orthogonal=through B](A,B)    \tkzGetPoint{b}
  \tkzInterLL(C,c)(B,b) \tkzGetPoint{T}
  \tkzInterLL(A,T)(C,D) \tkzGetPoint{P}
  \tkzDrawArc(I,B)(A)
  \tkzDrawSegments(A,B A,T C,D I,C) \tkzDrawSegment[new](I,C)
  \tkzDrawLine[add = 1 and 0](C,T)   \tkzDrawLine[add = 0 and 1](B,T)
  \tkzMarkRightAngle(I,C,T)
  \tkzDrawPoints(A,B,I,D,C,T)
  \tkzLabelPoints(A,B,I,D)  \tkzLabelPoints[above right](C,T)
  \tkzMarkSegment[pos=.25,mark=s|](C,D) \tkzMarkSegment[pos=.75,mark=s|](C,D)
\end{tikzpicture}
```

46.3.1. Square and rectangle of same area; Golden section

Book II, proposition XI _Euclid's Elements_

To construct Square and rectangle of same area.

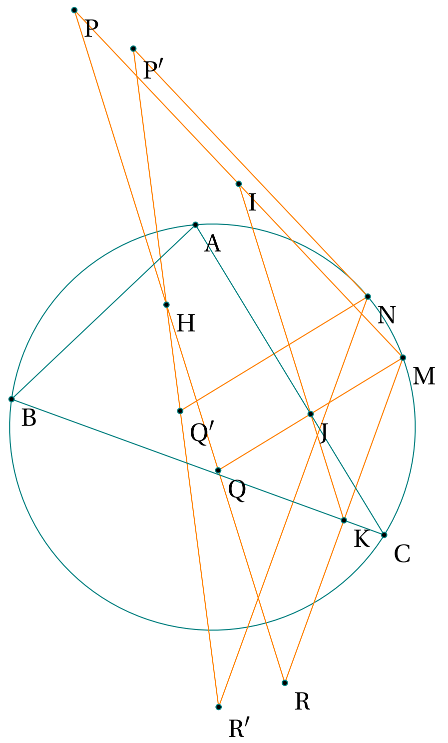



```
\begin{tikzpicture}[scale=.75]
 \tkzDefPoint(0,0){D} \tkzDefPoint(8,0){A}
 \tkzDefSquare(D,A) \tkzGetPoints{B}{C}
 \tkzDefMidPoint(D,A) \tkzGetPoint{E}
 \tkzInterLC(D,A)(E,B)\tkzGetSecondPoint{F}
 \tkzInterLC(A,B)(A,F)\tkzGetSecondPoint{G}
 \tkzDefSquare(A,F)\tkzGetFirstPoint{H}
 \tkzInterLL(C,D)(H,G)\tkzGetPoint{I}
 \tkzFillPolygon[teal!10](I,G,B,C)
 \tkzFillPolygon[teal!10](A,F,H,G)
 \tkzDrawArc[angles](E,B)(0,120)
 \tkzDrawSemiCircle(A,F)
 \tkzDrawSegments(A,F E,B H,I F,H)
 \tkzDrawPolygons(A,B,C,D)
 \tkzDrawPoints(A,...,I)
 \tkzLabelPoints(A,...,I)
\end{tikzpicture}
```

46.3.2. Steiner Line and Simson Line

Steiner Line and Simson Line

Consider the triangle ABC and a point M on its circumcircle. The projections of M on the sides of the triangle are on a line (Steiner Line), The three closest points to M on lines AB, AC, and BC are collinear. It's the Simson Line.

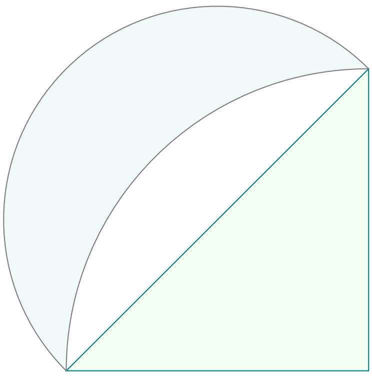



```
\begin{tikzpicture}[scale=.75,rotate=-20]
  \tkzDefPoint(0,0){B}
  \tkzDefPoint(2,4){A} \tkzDefPoint(7,0){C}
  \tkzDefCircle[circum](A,B,C)
  \tkzGetPoint{O}
  \tkzDrawCircle(O,A)
  \tkzCalcLength(O,A)
  \tkzGetLength{rOA}
  \tkzDefShiftPoint[O](40:\rOA){M}
  \tkzDefShiftPoint[O](60:\rOA){N}
  \tkzDefTriangleCenter[orthic](A,B,C)
  \tkzGetPoint{H}
  \tkzDefSpcTriangle[orthic,name=H](A,B,C){a,b,c}
  \tkzDefPointsBy[reflection=over A--B](M,N){P,P'}
  \tkzDefPointsBy[reflection=over A--C](M,N){Q,Q'}
  \tkzDefPointsBy[reflection=over C--B](M,N){R,R'}
  \tkzDefMidPoint(M,P)\tkzGetPoint{I}
  \tkzDefMidPoint(M,Q)\tkzGetPoint{J}
  \tkzDefMidPoint(M,R)\tkzGetPoint{K}
  \tkzDrawSegments[new](P,R M,P M,Q M,R N,P'%
   N,Q' N,R' P',R' I,K)
  \tkzDrawPolygons(A,B,C)
  \tkzDrawPoints(A,B,C,H,M,N,P,Q,R,P',Q',R',I,J,K)
  \tkzLabelPoints(A,B,C,H,M,N,P,Q,R,P',Q',R',I,J,K)
\end{tikzpicture}
```

46.4. Lune of Hippocrates

Lune of Hippocrates

From wikipedia : In geometry, the lune of Hippocrates, named after Hippocrates of Chios, is a lune bounded by arcs of two circles, the smaller of which has as its diameter a chord spanning a right angle on the larger circle.In the first figure, the area of the lune is equal to the area of the triangle ABC. Hippocrates of Chios (ancient Greek mathematician,)

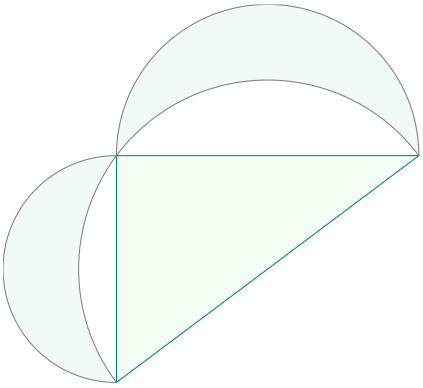


```
\begin{tikzpicture}
 \tkzInit[xmin=-2,xmax=5,ymin=-1,ymax=6]
 \tkzClip % allows you to define a bounding box
   % large enough
  \tkzDefPoint(0,0){A}\tkzDefPoint(4,0){B}
  \tkzDefSquare(A,B)
  \tkzGetFirstPoint{C}
  \tkzDrawPolygon[fill=green!5](A,B,C)
   \begin{scope}
     \tkzClipCircle[out](B,A)
     \tkzDrawSemiCircle[diameter,fill=teal!5](A,C)
   \end{scope}
   \tkzDrawArc[delta=0](B,C)(A)
\end{tikzpicture}
```

46.5. Lunes of Hasan Ibn al-Haytham

Lune of Hippocrates

From wikipedia : the Arab mathematician Hasan Ibn al-Haytham (Latinized name Alhazen) showed that two lunes, formed on the two sides of a right triangle, whose outer boundaries are semicircles and whose inner boundaries are formed by the circumcircle of the triangle, then the areas of these two lunes added together are equal to the area of the triangle. The lunes formed in this way from a right triangle are known as the lunes of Alhazen.

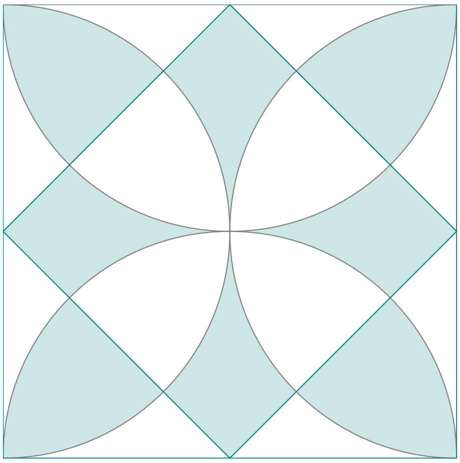


```
\begin{tikzpicture}[scale=.5,rotate=180]
  \tkzInit[xmin=-1,xmax=11,ymin=-4,ymax=7]
  \tkzClip
  \tkzDefPoints{0/0/A,8/0/B}
  \tkzDefTriangle[pythagore](A,B)
  \tkzGetPoint{C}
  \tkzDrawPolygon[fill=green!5](A,B,C)
  \tkzDefMidPoint(C,A) \tkzGetPoint{I}
  \begin{scope}
    \tkzClipCircle[out](I,A)
    \tkzDrawSemiCircle[diameter,fill=teal!5](B,A)
    \tkzDrawSemiCircle[diameter,fill=teal!5](C,B)
  \end{scope}
  \tkzSetUpCompass[/tkzcompass/delta=0]
  \tkzDrawSemiCircle[diameter](C,A)
\end{tikzpicture}
```

46.6. About clipping circles

About clipping circles

The problem is the management of the bounding box. First you have to define a rectangle in which the figure will be inserted. This is done with the first two lines.



```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=6,ymin=0,ymax=6]
  \tkzClip
  \tkzDefPoints{0/0/A, 6/0/B}
  \tkzDefSquare(A,B)      \tkzGetPoints{C}{D}
  \tkzDefMidPoint(A,B)        \tkzGetPoint{M}
  \tkzDefMidPoint(A,D)        \tkzGetPoint{N}
  \tkzDefMidPoint(B,C)        \tkzGetPoint{O}
  \tkzDefMidPoint(C,D)        \tkzGetPoint{P}
 \begin{scope}
  \tkzClipCircle[out](M,B) \tkzClipCircle[out](P,D)
  \tkzFillPolygon[teal!20](M,N,P,O)
 \end{scope}
 \begin{scope}
   \tkzClipCircle[out](N,A) \tkzClipCircle[out](O,C)
   \tkzFillPolygon[teal!20](M,N,P,O)
 \end{scope}
 \begin{scope}
   \tkzClipCircle(P,C) \tkzClipCircle(N,A)
   \tkzFillPolygon[teal!20](N,P,D)
 \end{scope}
 \begin{scope}
     \tkzClipCircle(O,C) \tkzClipCircle(P,C)
     \tkzFillPolygon[teal!20](P,C,O)
 \end{scope}
 \begin{scope}
     \tkzClipCircle(M,B)  \tkzClipCircle(O,B)
     \tkzFillPolygon[teal!20](O,B,M)
 \end{scope}
 \begin{scope}
     \tkzClipCircle(N,A) \tkzClipCircle(M,A)
     \tkzFillPolygon[teal!20](A,M,N)
 \end{scope}
 \tkzDrawSemiCircles(M,B N,A O,C P,D)
 \tkzDrawPolygons(A,B,C,D M,N,P,O)
 \end{tikzpicture}
```

46.7. Similar isosceles triangles

Similar isosceles triangles

*The following is from the excellent site **Descartes et les Mathématiques**. I did not modify the text and I am only the author of the programming of the figures.* http://debart.pagesperso-orange.fr/seconde/triangle.html

The following is from the excellent site **Descartes et les Mathématiques**. I did not modify the text and I am only the author of the programming of the figures.
http://debart.pagesperso-orange.fr/seconde/triangle.html

Bibliography:

- Géométrie au Bac - Tangente, special issue no. 8 - Exercise 11, page 11
- Elisabeth Busser and Gilles Cohen: 200 nouveaux problèmes du "Monde" - POLE 2007 (200 new problems of "Le Monde")
- Affaire de logique n° 364 - Le Monde February 17, 2004

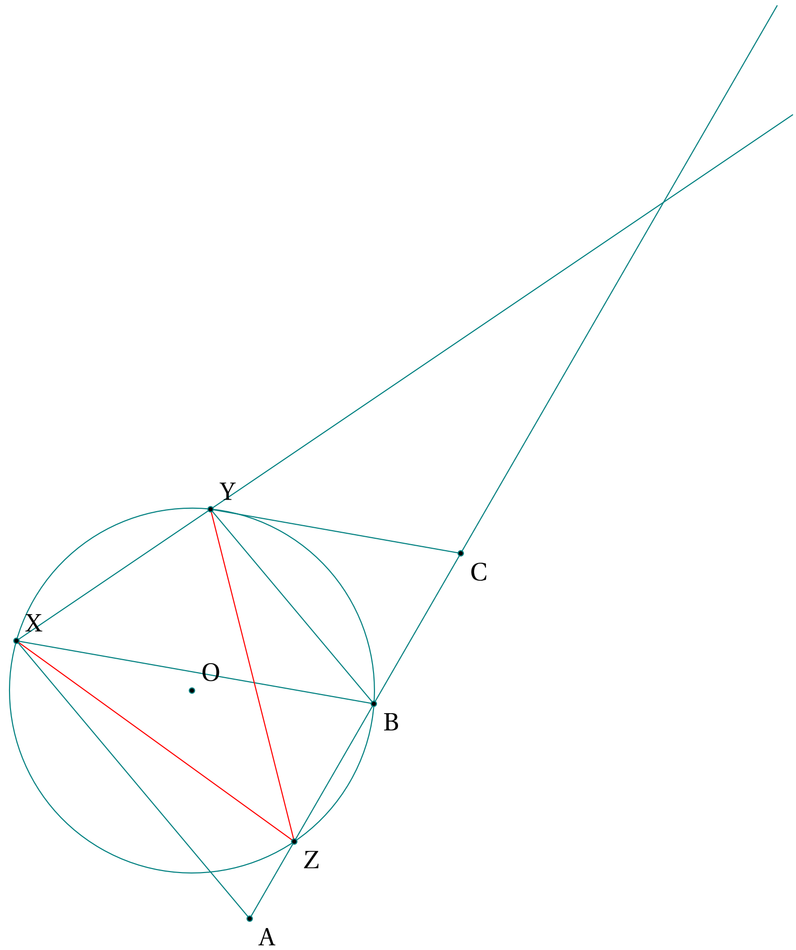
Two statements were proposed, one by the magazine *Tangente* and the other by *Le Monde*.

Editor of the magazine "Tangente": Two similar isosceles triangles AXB and BYC are constructed with main vertices X and Y, such that A, B and C are aligned and that these triangles are "indirect". Let α be the angle at vertex $\widehat{AXB} = \widehat{BYC}$. We then construct a third isosceles triangle XZY similar to the first two, with main vertex Z and "indirect". We ask to demonstrate that point Z belongs to the straight line (AC).

Editor of "Le Monde": We construct two similar isosceles triangles AXB and BYC with principal vertices X and Y, such that A, B and C are aligned and that these triangles are "indirect". Let α be the angle at vertex $\widehat{AXB} = \widehat{BYC}$. The point Z of the line segment [AC] is equidistant from the two vertices X and Y.
At what angle does he see these two vertices?

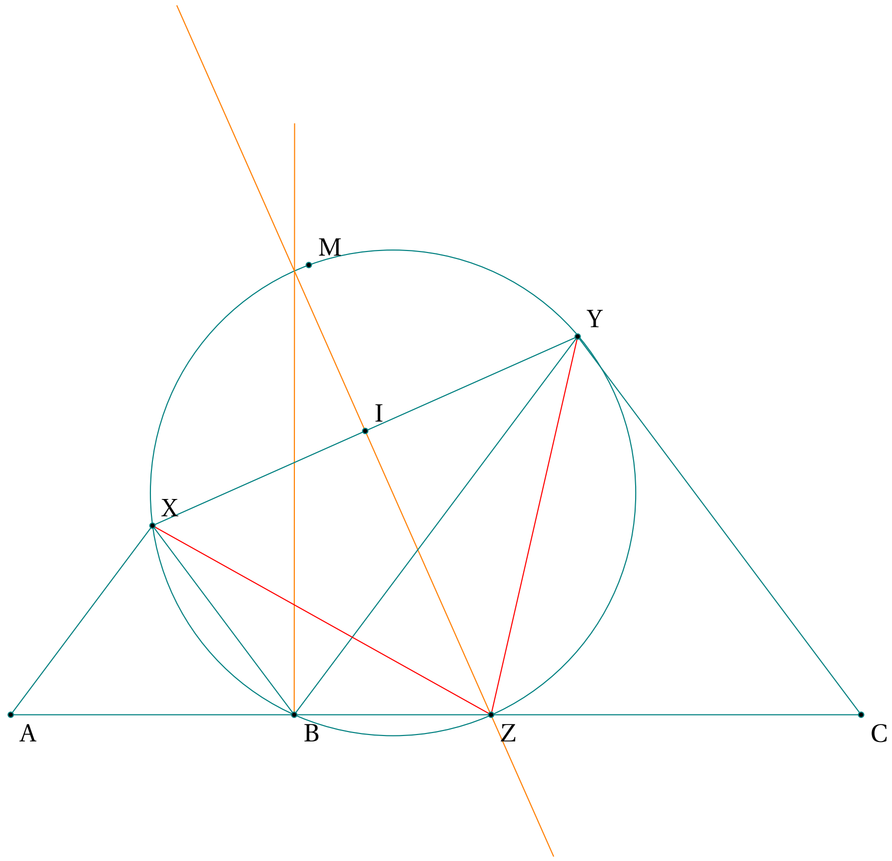
The constructions and their associated codes are on the next two pages, but you can search before looking. The programming respects (it seems to me ...) my reasoning in both cases.

46.8. Revised version of "Tangente"




```
\begin{tikzpicture}[scale=.8,rotate=60]
  \tkzDefPoint(6,0){X}   \tkzDefPoint(3,3){Y}
  \tkzDefShiftPoint[X](-110:6){A}    \tkzDefShiftPoint[X](-70:6){B}
  \tkzDefShiftPoint[Y](-110:4.2){A'} \tkzDefShiftPoint[Y](-70:4.2){B'}
  \tkzDefPointBy[translation= from A' to B ](Y) \tkzGetPoint{Y}
  \tkzDefPointBy[translation= from A' to B ](B') \tkzGetPoint{C}
  \tkzInterLL(A,B)(X,Y) \tkzGetPoint{O}
  \tkzDefMidPoint(X,Y) \tkzGetPoint{I}
  \tkzDefPointWith[orthogonal](I,Y)
  \tkzInterLL(I,tkzPointResult)(A,B) \tkzGetPoint{Z}
  \tkzDefCircle[circum](X,Y,B) \tkzGetPoint{O}
  \tkzDrawCircle(O,X)
  \tkzDrawLines[add = 0 and 1.5](A,C) \tkzDrawLines[add = 0 and 3](X,Y)
  \tkzDrawSegments(A,X B,X B,Y C,Y)   \tkzDrawSegments[color=red](X,Z Y,Z)
  \tkzDrawPoints(A,B,C,X,Y,O,Z)
  \tkzLabelPoints(A,B,C,Z)   \tkzLabelPoints[above right](X,Y,O)
\end{tikzpicture}
```

46.9. "Le Monde" version

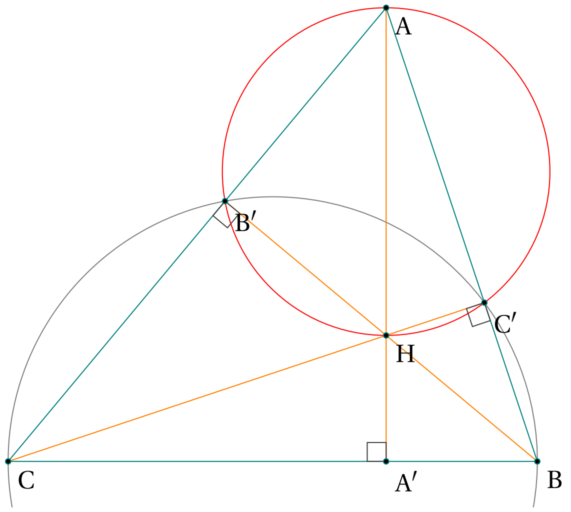



```
\begin{tikzpicture}[scale=1.25]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(3,0){B}
  \tkzDefPoint(9,0){C}
  \tkzDefPoint(1.5,2){X}
  \tkzDefPoint(6,4){Y}
  \tkzDefCircle[circum](X,Y,B) \tkzGetPoint{O}
  \tkzDefMidPoint(X,Y)              \tkzGetPoint{I}
  \tkzDefPointWith[orthogonal](I,Y)  \tkzGetPoint{i}
  \tkzDrawLines[add = 2 and 1,color=orange](I,i)
  \tkzInterLL(I,i)(A,B)              \tkzGetPoint{Z}
  \tkzInterLC(I,i)(O,B)              \tkzGetSecondPoint{M}
  \tkzDefPointWith[orthogonal](B,Z)  \tkzGetPoint{b}
  \tkzDrawCircle(O,B)
  \tkzDrawLines[add = 0 and 2,color=orange](B,b)
  \tkzDrawSegments(A,X B,X B,Y C,Y A,C X,Y)
  \tkzDrawSegments[color=red](X,Z Y,Z)
  \tkzDrawPoints(A,B,C,X,Y,Z,M,I)
  \tkzLabelPoints(A,B,C,Z)
  \tkzLabelPoints[above right](X,Y,M,I)
\end{tikzpicture}
```

46.10. Triangle altitudes

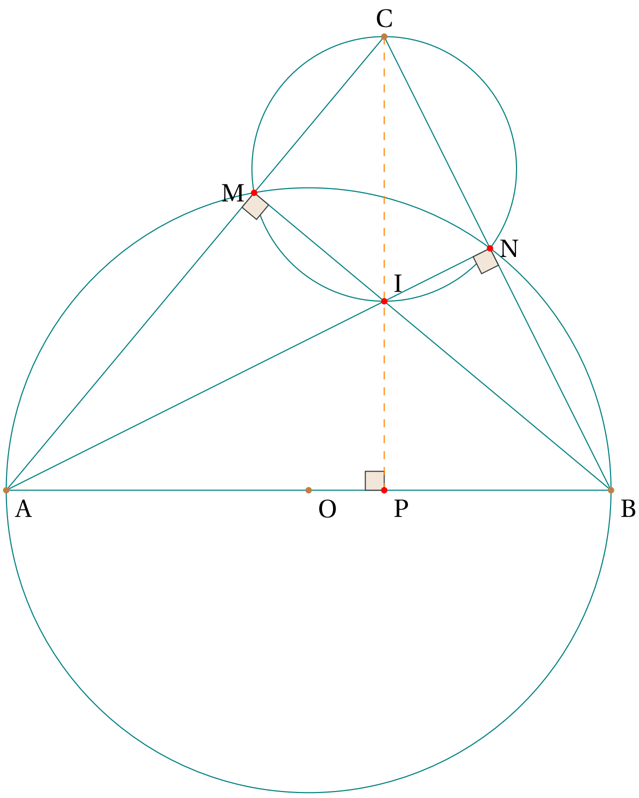
Triangle altitudes

From Wikipedia : The following is again from the excellent site ***Descartes et les Mathématiques*** *(Descartes and the Mathematics).* `http://debart.pagesperso-orange.fr/geoplan/geometrie_triangle.html`*. The three altitudes of a triangle intersect at the same H-point.*




```
\begin{tikzpicture}
   \tkzDefPoint(0,0){C} \tkzDefPoint(7,0){B}
   \tkzDefPoint(5,6){A}
   \tkzDefMidPoint(C,B) \tkzGetPoint{I}
   \tkzInterLC(A,C)(I,B)
   \tkzGetSecondPoint{B'}
   \tkzInterLC(A,B)(I,B)
   \tkzGetFirstPoint{C'}
   \tkzInterLL(B,B')(C,C') \tkzGetPoint{H}
   \tkzInterLL(A,H)(C,B) \tkzGetPoint{A'}
   \tkzDefCircle[circum](A,B',C') \tkzGetPoint{O}
   \tkzDrawArc(I,B)(C)
   \tkzDrawPolygon(A,B,C)
   \tkzDrawCircle[color=red](O,A)
   \tkzDrawSegments[color=orange](B,B' C,C' A,A')
   \tkzMarkRightAngles(C,B',B B,C',C C,A',A)
   \tkzDrawPoints(A,B,C,A',B',C',H)
   \tkzLabelPoints(A,B,C,A',B',C',H)
\end{tikzpicture}
```

46.11. Altitudes - other construction

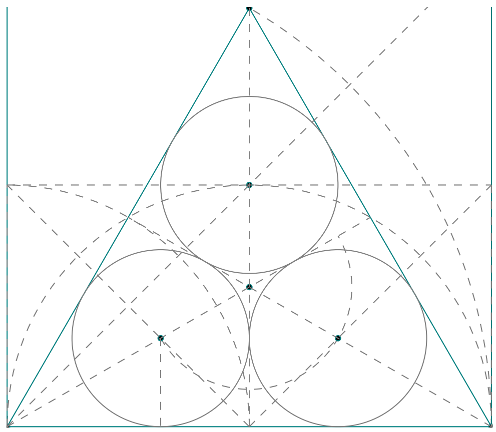



```
\begin{tikzpicture}
\tkzDefPoint(0,0){A} \tkzDefPoint(8,0){B}
\tkzDefPoint(5,6){C}
\tkzDefMidPoint(A,B)\tkzGetPoint{O}
\tkzDefPointBy[projection=onto A--B](C) \tkzGetPoint{P}
\tkzInterLC(C,A)(O,A)
\tkzGetSecondPoint{M}
\tkzInterLC(C,B)(O,A)
\tkzGetFirstPoint{N}
\tkzInterLL(B,M)(A,N)\tkzGetPoint{I}
\tkzDrawCircles[diameter](A,B I,C)
\tkzDrawSegments(C,A C,B A,B B,M A,N)
\tkzMarkRightAngles[fill=brown!20](A,M,B A,N,B A,P,C)
\tkzDrawSegment[style=dashed,color=orange](C,P)
\tkzLabelPoints(O,A,B,P)
\tkzLabelPoint[left](M){$M$}
\tkzLabelPoint[right](N){$N$}
\tkzLabelPoint[above](C){$C$}
\tkzLabelPoint[above right](I){$I$}
\tkzDrawPoints[color=red](M,N,P,I)
\tkzDrawPoints[color=brown](O,A,B,C)
\end{tikzpicture}
```

46.12. Three circles in an Equilateral Triangle

Three circles in an Equilateral Triangle

From Wikipedia : In geometry, the Malfatti circles are three circles inside a given triangle such that each circle is tangent to the other two and to two sides of the triangle. They are named after Gian Francesco Malfatti, who made early studies of the problem of constructing these circles in the mistaken belief that they would have the largest possible total area of any three disjoint circles within the triangle. Below is a study of a particular case with an equilateral triangle and three identical circles.

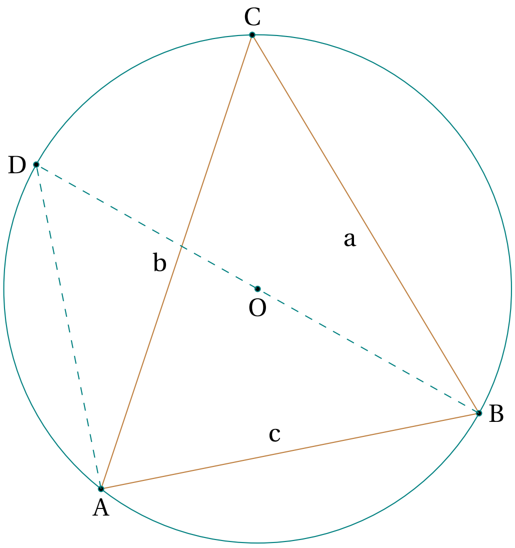


```
\begin{tikzpicture}[scale=.8]
  \tkzDefPoints{0/0/A,8/0/B,0/4/a,8/4/b,8/8/c}
  \tkzDefTriangle[equilateral](A,B) \tkzGetPoint{C}
  \tkzDefMidPoint(A,B) \tkzGetPoint{M}
  \tkzDefMidPoint(B,C) \tkzGetPoint{N}
  \tkzDefMidPoint(A,C) \tkzGetPoint{P}
  \tkzInterLL(A,N)(M,a) \tkzGetPoint{Ia}
  \tkzDefPointBy[projection = onto A--B](Ia)
  \tkzGetPoint{ha}
  \tkzInterLL(B,P)(M,b) \tkzGetPoint{Ib}
  \tkzDefPointBy[projection = onto A--B](Ib)
  \tkzGetPoint{hb}
  \tkzInterLL(A,c)(M,C) \tkzGetPoint{Ic}
  \tkzDefPointBy[projection = onto A--C](Ic)
  \tkzGetPoint{hc}
  \tkzInterLL(A,Ia)(B,Ib) \tkzGetPoint{G}
  \tkzDefSquare(A,B) \tkzGetPoints{D}{E}
  \tkzDrawPolygon(A,B,C)
  \tkzClipBB
  \tkzDrawSemiCircles[gray,dashed](M,B A,M
  A,B B,A G,Ia)
  \tkzDrawCircles[gray](Ia,ha Ib,hb Ic,hc)
  \tkzDrawPolySeg(A,E,D,B)
  \tkzDrawPoints(A,B,C,G,Ia,Ib,Ic)
  \tkzDrawSegments[gray,dashed](C,M A,N B,P
   M,a M,b A,a a,b b,B A,D Ia,ha)
\end{tikzpicture}
```

46.13. Law of sines

Law of sines

From wikipedia : *In trigonometry, the law of sines, sine law, sine formula, or sine rule is an equation relating the lengths of the sides of a triangle (any shape) to the sines of its angles.*




```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,5/1/B,2/6/C}
\tkzDefTriangleCenter[circum](A,B,C)
 \tkzGetPoint{O}
\tkzDefPointBy[symmetry= center O](B)
 \tkzGetPoint{D}
\tkzDrawPolygon[color=brown](A,B,C)
\tkzDrawCircle(O,A)
\tkzDrawPoints(A,B,C,D,O)
\tkzDrawSegments[dashed](B,D A,D)
\tkzLabelPoint[left](D){$D$}
\tkzLabelPoint[below](A){$A$}
\tkzLabelPoint[above](C){$C$}
\tkzLabelPoint[right](B){$B$}
\tkzLabelPoint[below](O){$O$}
\tkzLabelSegment(B,C){$a$}
\tkzLabelSegment[left](A,C){$b$}
\tkzLabelSegment(A,B){$c$}
\end{tikzpicture}
```

In the triangle ABC

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} \tag{1}$$

$$\widehat{C} = \widehat{D}$$

$$\frac{c}{2R} = \sin D = \sin C \tag{2}$$

Then

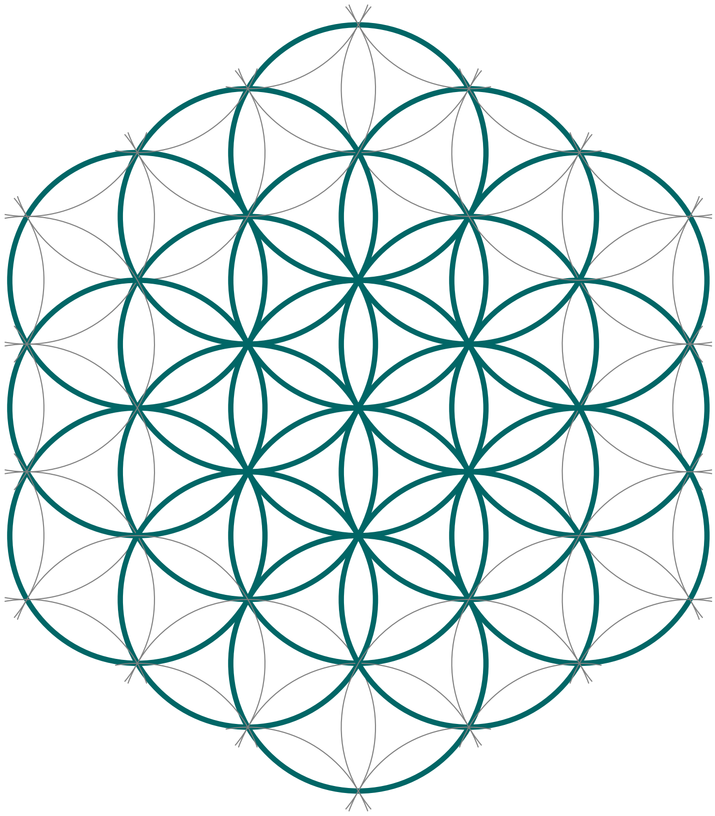
$$\frac{c}{\sin C} = 2R$$

46.14. Flower of Life

Book IV, proposition XI _Euclid's Elements_

Sacred geometry can be described as a belief system attributing a religious or cultural value to many of the fundamental forms of space and time. According to this belief system, the basic patterns of existence are perceived as sacred because in contemplating them one is contemplating the origin of all things. By studying the nature of these forms and their relationship to each other, one may seek to gain insight into the scientific, philosophical, psychological, aesthetic and mystical laws of the universe. The Flower of Life is considered to be a symbol of sacred geometry, said to contain ancient, religious value depicting the fundamental forms of space and time. In this sense, it is a visual expression of the connections life weaves through all mankind, believed by some to contain a type of Akashic Record of basic information of all living things.

One of the beautiful arrangements of circles found at the Temple of Osiris at Abydos, Egypt (Rawles 1997). Weisstein, Eric W. "Flower of Life." From MathWorld–A Wolfram Web Resource.
http://mathworld.wolfram.com/FlowerofLife.html



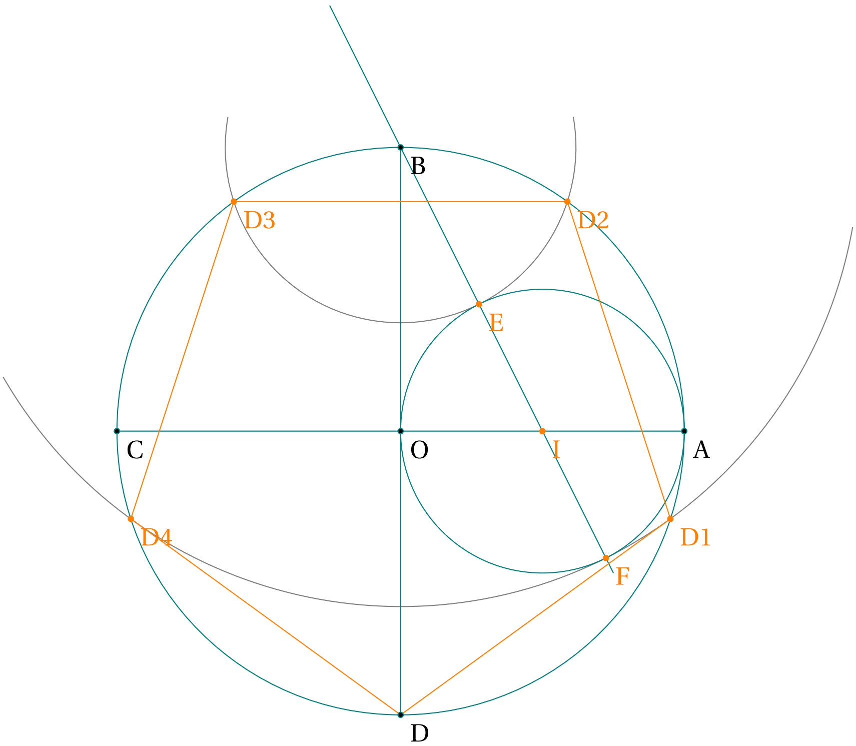
```
\begin{tikzpicture}[scale=.75]
  \tkzSetUpLine[line width=2pt,color=teal!80!black]
  \tkzSetUpCompass[line width=2pt,color=teal!80!black]
   \tkzDefPoint(0,0){O}  \tkzDefPoint(2.25,0){A}
   \tkzDrawCircle(O,A)
\foreach \i in {0,...,5}{
   \tkzDefPointBy[rotation= center O angle 30+60*\i](A)\tkzGetPoint{a\i}
   \tkzDefPointBy[rotation= center {a\i} angle  120](O)\tkzGetPoint{b\i}
   \tkzDefPointBy[rotation= center {a\i} angle  180](O)\tkzGetPoint{c\i}
   \tkzDefPointBy[rotation= center {c\i} angle  120](a\i)\tkzGetPoint{d\i}
   \tkzDefPointBy[rotation= center {c\i} angle   60](d\i)\tkzGetPoint{f\i}
   \tkzDefPointBy[rotation= center {d\i} angle   60](b\i)\tkzGetPoint{e\i}
   \tkzDefPointBy[rotation= center {f\i} angle   60](d\i)\tkzGetPoint{g\i}
   \tkzDefPointBy[rotation= center {d\i} angle   60](e\i)\tkzGetPoint{h\i}
   \tkzDefPointBy[rotation= center {e\i} angle  180](b\i)\tkzGetPoint{k\i}
   \tkzDrawCircle(a\i,O)
   \tkzDrawCircle(b\i,a\i)
   \tkzDrawCircle(c\i,a\i)
   \tkzDrawArc[rotate](f\i,d\i)(-120)
   \tkzDrawArc[rotate](e\i,d\i)(180)
   \tkzDrawArc[rotate](d\i,f\i)(180)
   \tkzDrawArc[rotate](g\i,f\i)(60)
   \tkzDrawArc[rotate](h\i,d\i)(60)
   \tkzDrawArc[rotate](k\i,e\i)(60)
}
   \tkzClipCircle(O,f0)
\end{tikzpicture}
```

46.15. Pentagon in a circle

Book IV, proposition XI _Euclid's Elements_

To inscribe an equilateral and equiangular pentagon in a given circle.

```
\begin{tikzpicture}
  \tkzDefPoint(0,0){O} \tkzDefPoint(5,0){A}
  \tkzDefPoint(0,5){B} \tkzDefPoint(-5,0){C}
  \tkzDefPoint(0,-5){D}
  \tkzDefMidPoint(A,O)      \tkzGetPoint{I}
  \tkzInterLC(I,B)(I,A)     \tkzGetPoints{F}{E}
  \tkzInterCC(O,C)(B,E)     \tkzGetPoints{D3}{D2}
  \tkzInterCC(O,C)(B,F)     \tkzGetPoints{D4}{D1}
  \tkzDrawArc[angles](B,E)(180,360)
  \tkzDrawArc[angles](B,F)(220,340)
  \tkzDrawLine[add=.5 and .5](B,I)
  \tkzDrawCircle(O,A)
  \tkzDrawCircle[diameter](O,A)
  \tkzDrawSegments(B,D C,A)
  \tkzDrawPolygon[new](D,D1,D2,D3,D4)
  \tkzDrawPoints(A,...,D,O)
  \tkzDrawPoints[new](E,F,I,D1,D2,D4,D3)
  \tkzLabelPoints(A,...,D,O)
  \tkzLabelPoints[new](I,E,F,D1,D2,D4,D3)
 \end{tikzpicture}
```

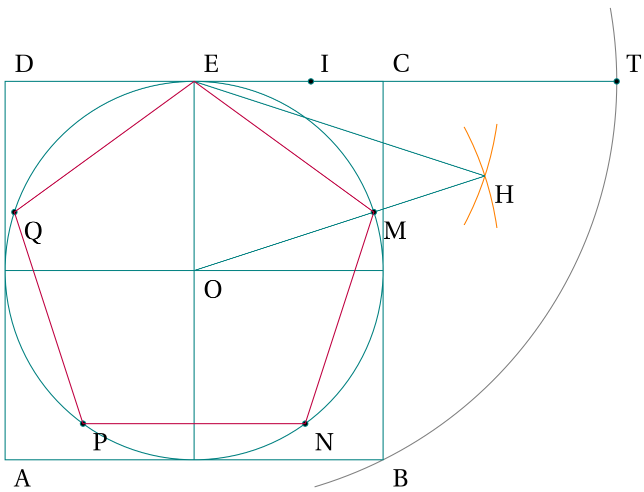


46.16. Pentagon in a square

Pentagon in a square

: *To inscribe an equilateral and equiangular pentagon in a given square.*

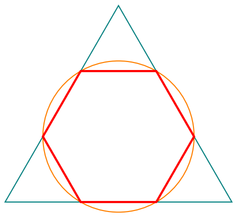
```
  \begin{tikzpicture}
  \tkzDefPoint(-5,-5){A} \tkzDefPoint(0,0){O}
  \tkzDefPoint(+5,-5){B} \tkzDefPoint(0,-5){F}
  \tkzDefPoint(+5,0){F'} \tkzDefPoint(0,+5){E} \tkzDefPoint(-5,0){K}
  \tkzDefSquare(A,B)                         \tkzGetPoints{C}{D}
  \tkzInterLC(D,C)(E,B)                      \tkzGetSecondPoint{T}
  \tkzDefMidPoint(D,T)                       \tkzGetPoint{I}
  \tkzInterCC[with nodes](O,D,I)(E,D,I)      \tkzGetSecondPoint{H}
  \tkzInterLC(O,H)(O,E)                      \tkzGetSecondPoint{M}
  \tkzInterCC(O,E)(E,M)                      \tkzGetFirstPoint{Q}
  \tkzInterCC[with nodes](O,O,E)(Q,E,M)      \tkzGetFirstPoint{P}
  \tkzInterCC[with nodes](O,O,E)(P,E,M)      \tkzGetFirstPoint{N}
  \tkzCompass(O,H)
  \tkzCompass(E,H)
  \tkzDrawArc(E,B)(T)
  \tkzDrawPolygon(A,B,C,D)
  \tkzDrawCircle(O,E)
  \tkzDrawSegments[new](T,I O,H E,H E,F F',K)
  \tkzDrawPoints(T,M,Q,P,N,I)
  \tkzDrawPolygon[new](M,E,Q,P,N)
  \tkzLabelPoints(A,B,O,N,P,Q,M,H)
  \tkzLabelPoints[above right](C,D,E,I,T)
\end{tikzpicture}
```



46.17. Hexagon Inscribed

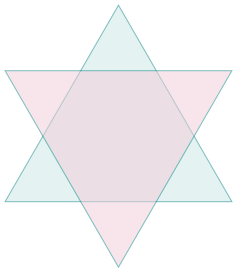
Hexagon Inscribed

To inscribe a regular hexagon in a given equilateral triangle perfectly inside it (no boarders).



```
\begin{tikzpicture}[scale=.5]
 \pgfmathsetmacro{\c}{6}
 \tkzDefPoints{0/0/A,\c/0/B}
 \tkzDefTriangle[equilateral](A,B)\tkzGetPoint{C}
 \tkzDefTriangleCenter[centroid](A,B,C)
 \tkzGetPoint{I}
 \tkzDefPointBy[homothety=center A ratio 1./3](B)
 \tkzGetPoint{c1}
 \tkzInterLC(B,C)(I,c1) \tkzGetPoints{a1}{a2}
 \tkzInterLC(A,C)(I,c1) \tkzGetPoints{b1}{b2}
 \tkzInterLC(A,B)(I,c1) \tkzGetPoints{c1}{c2}
 \tkzDrawPolygon(A,B,C)
 \tkzDrawCircle[thin,orange](I,c1)
 \tkzDrawPolygon[red,thick](a2,a1,b2,b1,c2,c1)
\end{tikzpicture}
```

Another solution

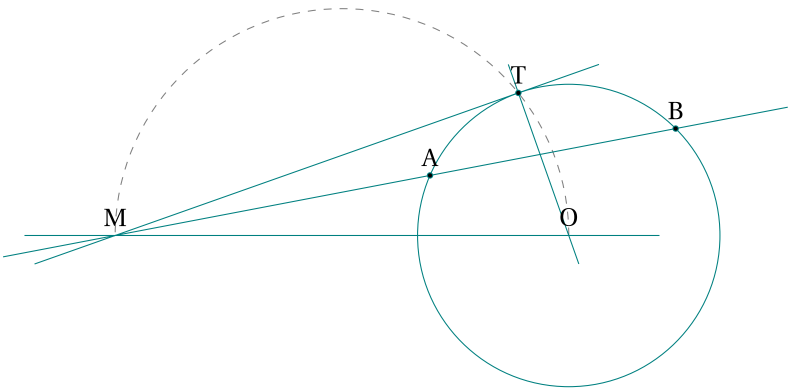


```
\begin{tikzpicture}[scale=.5]
 \pgfmathsetmacro{\c}{6}
 \tkzDefPoints{0/0/A,\c/0/B}
 \tkzDefTriangle[equilateral](A,B)\tkzGetPoint{C}
 \tkzDefTriangleCenter[centroid](A,B,C)
 \tkzGetPoint{I}
 \tkzDefPointsBy[rotation= center I%
                angle 60](A,B,C){a,b,c}
 \tkzDrawPolygon[fill=teal!20,opacity=.5](A,B,C)
 \tkzDrawPolygon[fill=purple!20,opacity=.5](a,b,c)
\end{tikzpicture}
```

46.18. Power of a point with respect to a circle

Power of a point with respect to a circle

$\overline{\mathrm{MA}} \times \overline{\mathrm{MB}} = \mathrm{MT}^2 = \mathrm{MO}^2 - \mathrm{OT}^2$

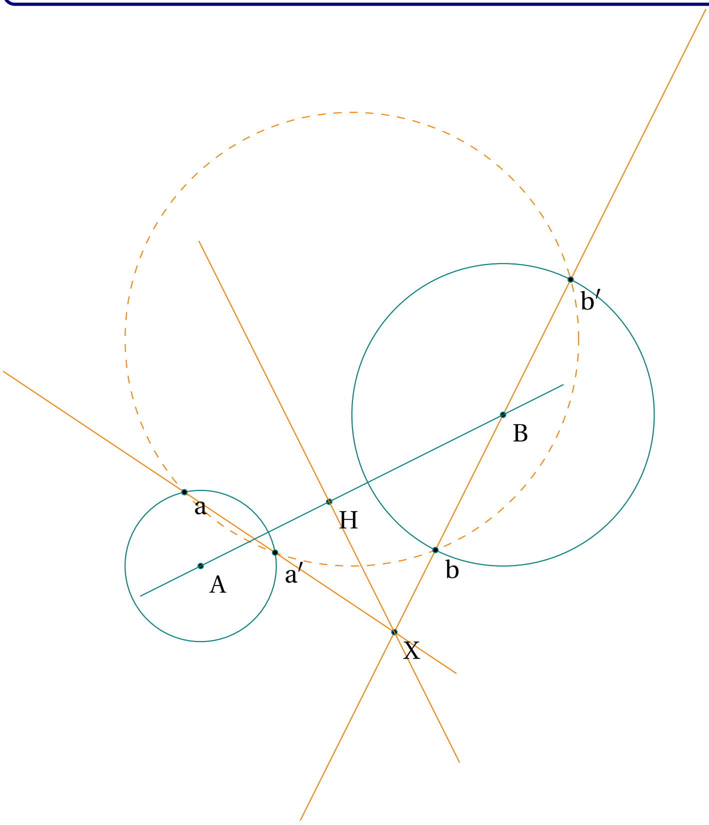



```
\begin{tikzpicture}
 \pgfmathsetmacro{\r}{2}%
 \pgfmathsetmacro{\xO}{6}%
 \pgfmathsetmacro{\xE}{\xO-\r}%
 \tkzDefPoints{0/0/M,\xO/0/O,\xE/0/E}
 \tkzDefCircle[diameter](M,O)
 \tkzGetPoint{I}
 \tkzInterCC(I,O)(O,E) \tkzGetPoints{T}{T'}
 \tkzDefShiftPoint[O](45:2){B}
 \tkzInterLC(M,B)(O,E) \tkzGetPoints{A}{B}
 \tkzDrawCircle(O,E)
 \tkzDrawSemiCircle[dashed](I,O)
 \tkzDrawLine(M,O)
 \tkzDrawLines(M,T O,T M,B)
 \tkzDrawPoints(A,B,T)
 \tkzLabelPoints[above](A,B,O,M,T)
\end{tikzpicture}
```

46.19. Radical axis of two non-concentric circles

Radical axis of two non-concentric circles

From Wikipedia : *In geometry, the radical axis of two non-concentric circles is the set of points whose power with respect to the circles are equal. For this reason the radical axis is also called the power line or power bisector of the two circles. The notation radical axis was used by the French mathematician M. Chasles as axe radical.*

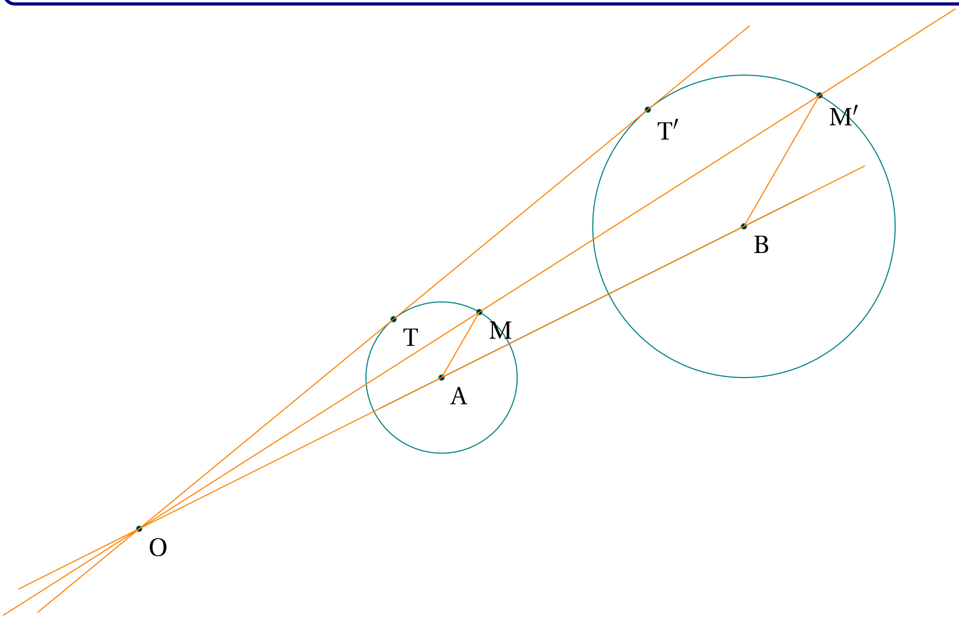



```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,4/2/B,2/3/K}
\tkzInterCC[R](A,1)(K,3) \tkzGetPoints{a}{a'}
\tkzInterCC[R](B,2)(K,3) \tkzGetPoints{b}{b'}
\tkzDrawLines[color=red,add=2 and 2](a,a')
\tkzDrawLines[color=red,add=1 and 1](b,b')
\tkzInterLL(a,a')(b,b') \tkzGetPoint{X}
\tkzDefPointBy[projection= onto A--B](X) \tkzGetPoint{H}
\tkzDrawCircle[R](A,1)\tkzDrawCircle[R](B,2)
\tkzDrawCircle[R,dashed,orange](K,3)
\tkzDrawPoints(A,B,H,X,a,b,a',b')
\tkzDrawLine(A,B)
\tkzDrawLine[add= 1 and 2](X,H)
\tkzLabelPoints(A,B,H,X,a,b,a',b')
\end{tikzpicture}
```

46.20. External homothetic center

External homothetic center

From Wikipedia : *Given two nonconcentric circles, draw radii parallel and in the same direction. Then the line joining the extremities of the radii passes through a fixed point on the line of centers which divides that line externally in the ratio of radii. This point is called the external homothetic center, or external center of similitude (Johnson 1929, pp. 19-20 and 41).*

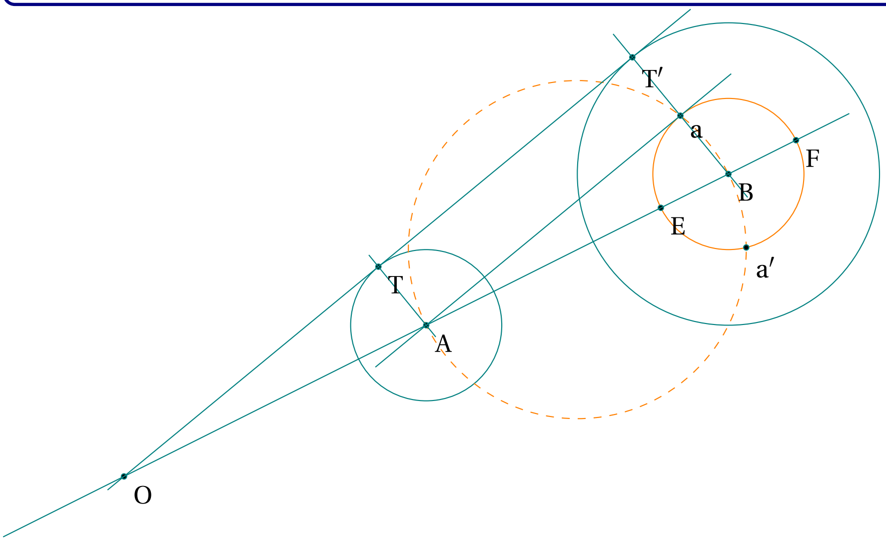



```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,4/2/B,2/3/K}
\tkzDefShiftPoint[A](60:1){M}
\tkzDefShiftPoint[B](60:2){M'}
\tkzInterLL(A,B)(M,M') \tkzGetPoint{O}
\tkzDefTangent[from = O](B,M') \tkzGetPoints{X}{T'}
\tkzDefTangent[from = O](A,M) \tkzGetPoints{X}{T}
\tkzDrawCircle[R](A,1)\tkzDrawCircle[R](B,2)
\tkzDrawLine(A,B)
\tkzDrawPoints(A,B,O,T,T',M,M')
\tkzDrawLines[new](O,B O,T' O,M')
\tkzDrawSegments[new](A,M B,M')
\tkzLabelPoints(A,B,O,T,T',M,M')
\end{tikzpicture}
```

46.21. Tangent lines to two circles

Tangent lines to two circles

For two circles, there are generally four distinct lines that are tangent to both if the two circles are outside each other. For two of these, the external tangent lines, the circles fall on the same side of the line; the external tangent lines intersect in the external homothetic center

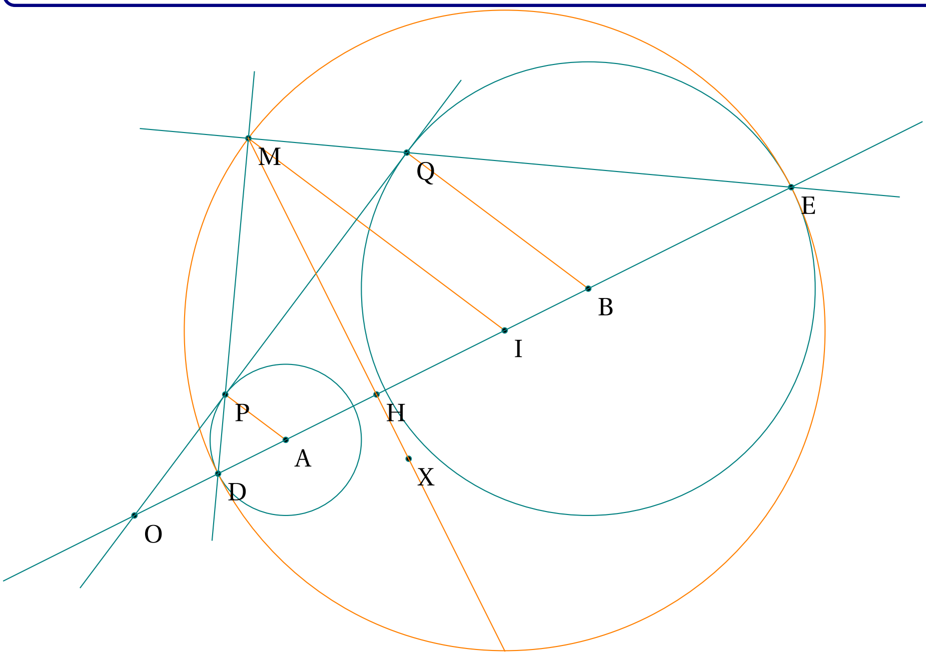



```
 \begin{tikzpicture}
\pgfmathsetmacro{\r}{1}%
\pgfmathsetmacro{\R}{2}%
\pgfmathsetmacro{\rt}{\R-\r}%
\tkzDefPoints{0/0/A,4/2/B,2/3/K}
\tkzDefMidPoint(A,B) \tkzGetPoint{I}
\tkzInterLC[R](A,B)(B,\rt) \tkzGetPoints{E}{F}
\tkzInterCC(I,B)(B,F) \tkzGetPoints{a}{a'}
\tkzInterLC[R](B,a)(B,\R) \tkzGetPoints{X'}{T'}
\tkzDefTangent[at=T'](B) \tkzGetPoint{h}
\tkzInterLL(T',h)(A,B) \tkzGetPoint{O}
\tkzInterLC[R](O,T')(A,\r) \tkzGetPoints{T}{T}
\tkzDrawCircle[R](A,\r)          \tkzDrawCircle[R](B,\R)
\tkzDrawCircle[R,orange](B,\rt)  \tkzDrawCircle[orange,dashed](I,B)
\tkzDrawPoints(O,A,B,a,a',E,F,T',T)
\tkzDrawLines(O,B A,a B,T' A,T)
\tkzDrawLines[add= 1 and 8](T',h)
\tkzLabelPoints(O,A,B,a,a',E,F,T,T')
 \end{tikzpicture}
```

46.22. Tangent lines to two circles with radical axis

Tangent lines to two circles with radical axis

As soon as two circles are not concentric, we can construct their radical axis, the set of points of equal power with respect to the two circles. We know that the radical axis is a line orthogonal to the line of the centers. Note that if we specify P *and* Q *as the points of contact of one of the common exterior tangents with the two circles and* D *and* E *as the points of the circles outside [AB], then (DP) and (EQ) intersect on the radical axis of the two circles. We will show that this property is always true and that it allows us to construct common tangents, even when the circles have the same radius.*




```
\begin{tikzpicture}
\tkzDefPoints{0/0/A,4/2/B,2/3/K}
\tkzDrawCircle[R](A,1)\tkzDrawCircle[R](B,3)
\tkzInterCC[R](A,1)(K,3) \tkzGetPoints{a}{a'}
\tkzInterCC[R](B,3)(K,3) \tkzGetPoints{b}{b'}
\tkzInterLL(a,a')(b,b') \tkzGetPoint{X}
\tkzDefPointBy[projection= onto A--B](X) \tkzGetPoint{H}
\tkzGetPoint{C}
\tkzInterLC[R](A,B)(B,3) \tkzGetPoints{b1}{E}
\tkzInterLC[R](A,B)(A,1) \tkzGetPoints{D}{a2}
\tkzDefMidPoint(D,E) \tkzGetPoint{I}
\tkzDrawCircle[orange](I,D)
\tkzInterLC(X,H)(I,D) \tkzGetPoints{M'}{M}
\tkzInterLC(M,D)(A,D) \tkzGetPoints{P'}{P}
\tkzInterLC(M,E)(B,E) \tkzGetPoints{Q}{Q'}
\tkzInterLL(P,Q)(A,B) \tkzGetPoint{O}
\tkzDrawSegments[orange](A,P I,M B,Q)
\tkzDrawPoints(A,B,D,E,M,I,O,P,Q,X,H)
\tkzDrawLines(O,E M,D M,E O,Q)
\tkzDrawLine[add= 3 and 4,orange](X,H)
\tkzLabelPoints(A,B,D,E,M,I,O,P,Q,X,H)
\end{tikzpicture}
```

46.23. Definition of a circle _Apollonius_

Definition of a circle _Apollonius_

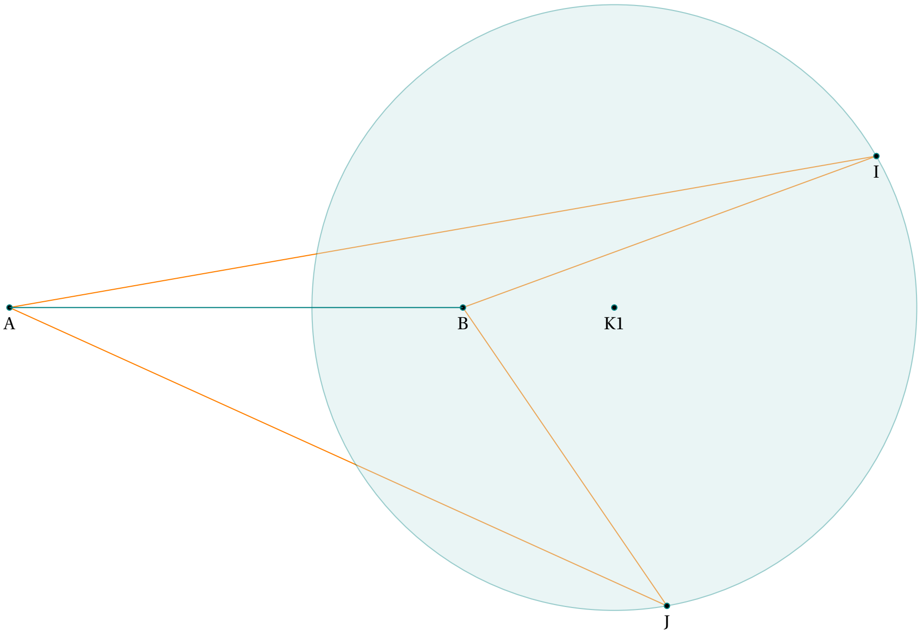
From Wikipedia : *Apollonius showed that a circle can be defined as the set of points in a plane that have a specified ratio of distances to two fixed points, known as foci. This Apollonian circle is the basis of the Apollonius pursuit problem. ... The solutions to this problem are sometimes called the circles of Apollonius.*

Explanation
A circle is the set of points in a plane that are equidistant from a given point O. The distance r from the center is called the radius, and the point O is called the center. It is the simplest definition but it is not the only one. Apollonius of Perga gives another definition : The set of all points whose distances from two fixed points are in a constant ratio is a circle.
With `tkz-euclide` is easy to show you the last definition

The code and the analyse

```
\documentclass{standalone}
    % Excellent class to show the result and to verify the bounding box.
\usepackage{tkz-euclide}
    % no need to use \usetkzobj !
\begin{document}
\begin{tikzpicture}[scale=1.5]
    % Firstly we defined two fixed point.
    % The figure depends of these points and the ratio K
\tkzDefPoint(0,0){A}
\tkzDefPoint(4,0){B}
    % tkz-euclide.sty knows about the apollonius's circle
    % with K=2 we search some points like  I such as IA=2 x IB
\tkzDefCircle[apollonius,K=2](A,B) \tkzGetPoint{K1}
\tkzGetLength{rAp}
\tkzDefPointOnCircle[angle=30,center=K1,radius=\rAp]
\tkzGetPoint{I}
\tkzDefPointOnCircle[angle=280,center=K1,radius=\rAp]
\tkzGetPoint{J}
\tkzDrawSegments[new](A,I I,B A,J J,B)
\tkzDrawCircle[R,color = teal,fill=MidnightBlue!20,opacity=.4](K1,\rAp pt)
\tkzDrawPoints(A,B,K1,I,J)
\tkzDrawSegment(A,B)
\tkzLabelPoints[below,font=\scriptsize](A,B,K1,I,J)
\end{tikzpicture}
\end{document}
```

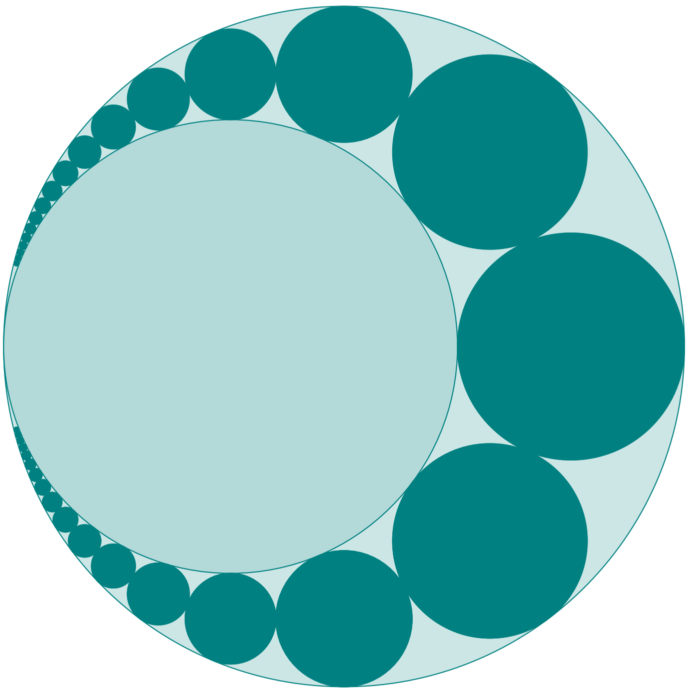
The result



46.24. Application of Inversion : Pappus chain

Pappus chain

From Wikipedia *In geometry, the Pappus chain is a ring of circles between two tangent circles investigated by Pappus of Alexandria in the 3rd century AD.*

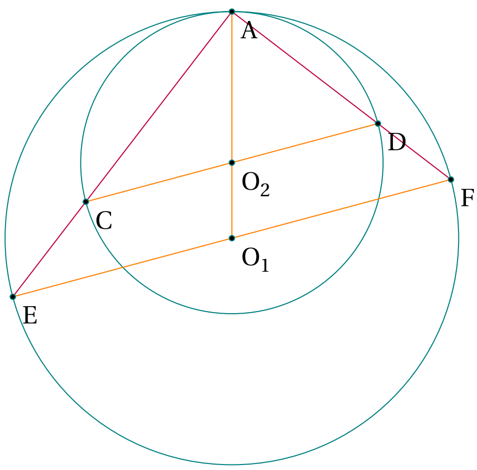


```
  \pgfmathsetmacro{\xB}{6}%
  \pgfmathsetmacro{\xC}{9}%
  \pgfmathsetmacro{\xD}{(\xC*\xC)/\xB}%
  \pgfmathsetmacro{\xJ}{(\xC+\xD)/2}%
  \pgfmathsetmacro{\r}{\xD-\xJ}%
  \pgfmathsetmacro{\nc}{16}%
\begin{tikzpicture}[ultra thin]
  \tkzDefPoints{0/0/A,\xB/0/B,\xC/0/C,\xD/0/D}
  \tkzDrawCircle[diameter,fill=teal!20](A,C)
  \tkzDrawCircle[diameter,fill=teal!30](A,B)
  \foreach \i in {-\nc,...,0,...,\nc}
  {\tkzDefPoint(\xJ,2*\r*\i){J}
   \tkzDefPoint(\xJ,2*\r*\i-\r){H}
   \tkzDefCircleBy[inversion = center A through C](J,H)
   \tkzDrawCircle[diameter,fill=teal](tkzFirstPointResult,tkzSecondPointResult)}
\end{tikzpicture}
```

46.25. Book of lemmas proposition 1 Archimedes

Book of lemmas proposition 1 Archimedes

If two circles touch at A, *and if* [CD], [EF] *be parallel diameters in them,* A, C *and* E *are aligned.*



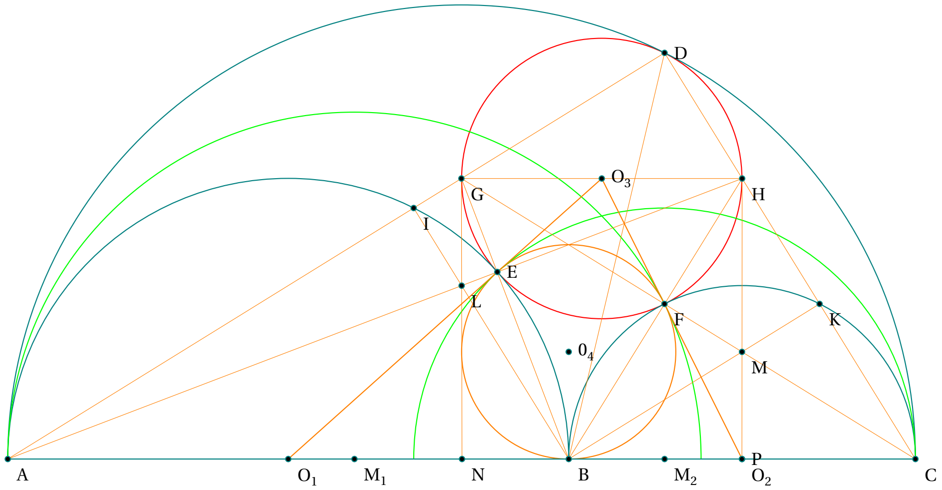

```
\begin{tikzpicture}
  \tkzDefPoints{0/0/O_1,0/1/O_2,0/3/A}
  \tkzDefPoint(15:3){F}
  \tkzInterLC(F,O_1)(O_1,A) \tkzGetSecondPoint{E}
  \tkzDefLine[parallel=through O_2](E,F)
  \tkzGetPoint{x}
  \tkzInterLC(x,O_2)(O_2,A) \tkzGetPoints{D}{C}
  \tkzDrawCircles(O_1,A O_2,A)
  \tkzDrawSegments[new](O_1,A 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}
```

(CD) $\parallel$ (EF) (AO_1) is secant to these two lines so $\widehat{A0_2C} = \widehat{A0_1E}$.
Since the triangles AO_2C and AO_1E are isosceles the angles at the base are equal widehatAC0_2 = $\widehat{AE0_1} = \widehat{CA0_2} = \widehat{EA0_1}$. Thus A,C and E are aligned

46.26. Book of lemmas proposition 6 Archimedes

Book of lemmas proposition 6 Archimedes

Let AC, *the diameter of a semicircle, be divided at* B *so that* AC/AB = ϕ *or in any ratio]. Describe semicircles within the first semicircle and on* AB, BC *as diameters, and suppose a circle drawn touching the all three semicircles. If* GH *be the diameter of this circle, to find relation between* GH *and* AC.



Let GH be the diameter of the circle which is parallel to AC, and let the circle touch the semicircles on AC, AB, BC in D, E, F respectively.
Then, by Prop. 1 A,G and D are aligned, ainsi que D, H and C.
For a like reason A E and H are aligned, C F and Gare aligned, as also are B E and G, B F and H.
Let (AD) meet the semicircle on [AC] at I, and let (BD) meet the semicircle on [BC] in K. Join CI, CK meeting AE, BF in L, M, and let GL, HM produced meet AB in N, P respectively.
Now, in the triangle AGB, the perpendiculars from A, C on the opposite sides meet in L. Therefore by the properties of triangles, (GN) is perpendicular to (AC). Similarly (HP) is perpendicular to (BC).
Again, since the angles at I, K, D are right, (CK) is parallel to (AD), and (CI) to (BD).
Therefore

$$\frac{AB}{BC} = \frac{AL}{LH} = \frac{AN}{NP} \quad \text{and} \quad \frac{BC}{AB} = \frac{CM}{MG} = \frac{PC}{NP}$$

hence

$$\frac{AN}{NP} = \frac{NP}{PC} \quad \text{so} \quad NP^2 = AN \times PC$$

Now suppose that B divides [AC] according to the divine proportion that is :

$$\phi = \frac{AB}{BC} = \frac{AC}{AB} \quad \text{then} \quad AN = \phi NP and NP = \phi PC$$

We have

$$AC = AN + NP + PC \quad \text{either} \quad AB + BC == AN + NP + PC \quad \text{or} \quad (\phi+1)BC = AN + NP + PC$$

we get

$$(\phi+1)BC = \phi NP + NP + PC = (\phi+1)NP + PC = \phi(\phi+1)PC + PC = \phi^2 + \phi + 1)PC$$

as

$$\phi^2 = \phi + 1 \quad \text{then} \quad (\phi+1)BC = 2(\phi+1)PC \quad \text{i.e.} \quad BC = 2PC$$

That is, p is the middle of the segment BC.
Part of the proof from `https://www.cut-the-knot.org`

46.27. "The" Circle of APOLLONIUS

The Apollonius circle of a triangle _Apollonius_

The circle which touches all three excircles of a triangle and encompasses them is often known as "the" Apollonius circle (Kimberling 1998, p. 102)

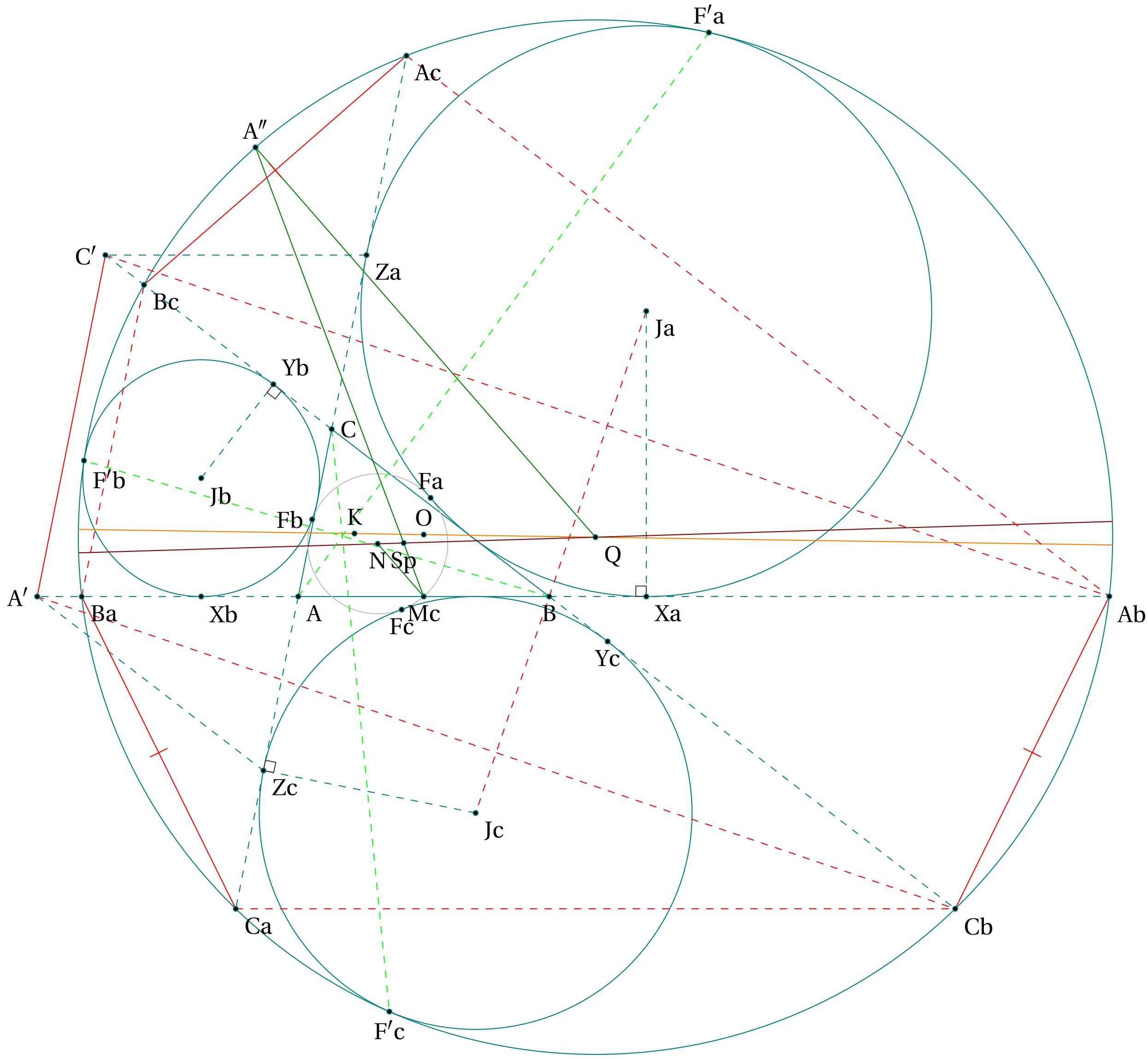
Explanation
The purpose of the first examples was to show the simplicity with which we could recreate these propositions. With TikZ you need to do calculations and use trigonometry while with `tkz-euclide` you only need to build simple objects
But don't forget that behind or far above `tkz-euclide` there is TikZ. I'm only creating an interface between TikZ and the user of my package.
The last example is very complex and it is to show you all that we can do with `tkz-euclide`.

The code and the analyse

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

\begin{tikzpicture}[scale=1]
\tkzDefPoints{0/0/A,6/0/B,0.8/4/C}
% we need some special points if the triangle,  tkz-euclide.sty knows about them

\tkzDefTriangleCenter[euler](A,B,C)    \tkzGetPoint{N} % or   \tkzEulerCenter(A,B,C)
\tkzDefTriangleCenter[circum](A,B,C)   \tkzGetPoint{O} %    \tkzCircumCenter(A,B,C)
\tkzDefTriangleCenter[lemoine](A,B,C)  \tkzGetPoint{K}
\tkzDefTriangleCenter[ortho](A,B,C)    \tkzGetPoint{H}

 % \tkzDefSpcTriangle new macro to define new triangle in relation wth ABC
\tkzDefSpcTriangle[excentral,name=J](A,B,C){a,b,c}
\tkzDefSpcTriangle[centroid,name=M](A,B,C){a,b,c}
\tkzDefCircle[in](Ma,Mb,Mc)             \tkzGetPoint{Sp} % Sp Spieker center

% here I used the definition but tkz-euclide knows this point
% \tkzDefTriangleCenter[spieker](A,B,C)    \tkzGetPoint{Sp}
% each center has three projections on the sides of the triangle ABC
% We can do this with one macro
\tkzDefProjExcenter[name=J](A,B,C)(a,b,c){Y,Z,X}

% but possible is
% \tkzDefPointBy[projection=onto A--C ](Ja)  \tkzGetPoint{Za}
\tkzDefLine[parallel=through Za](A,B)      \tkzGetPoint{Xc}
\tkzInterLL(Za,Xc)(C,B)                    \tkzGetPoint{C'}
\tkzDefLine[parallel=through Zc](B,C)      \tkzGetPoint{Ya}
\tkzInterLL(Zc,Ya)(A,B)                    \tkzGetPoint{A'}
\tkzDefPointBy[reflection= over Ja--Jc](C')\tkzGetPoint{Ab}
\tkzDefPointBy[reflection= over Ja--Jc](A')\tkzGetPoint{Cb}

% Now we can get the center of THE CIRCLE : Q
% BUT we need to find the radius or a point on the circle
\tkzInterLL(K,O)(N,Sp)                     \tkzGetPoint{Q}
\tkzInterLC(A,B)(Q,Cb)                     \tkzGetSecondPoint{Ba}
\tkzInterLC(A,C)(Q,Cb)                     \tkzGetPoints{Ca}{Ac}
\tkzInterLC(B,C')(Q,Cb)                    \tkzGetSecondPoint{Bc}
\tkzInterLC(Q,Ja)(Q,Cb)                    \tkzGetSecondPoint{F'a}
\tkzInterLC(Q,Jc)(Q,Cb)                    \tkzGetSecondPoint{F'c}
\tkzInterLC(Q,Jb)(Q,Cb)                    \tkzGetSecondPoint{F'b}
\tkzInterLC(Sp,F'a)(Ja,Za)                 \tkzGetFirstPoint{Fa}
\tkzInterLC(Sp,F'b)(Jb,Yb)                 \tkzGetFirstPoint{Fb}
```

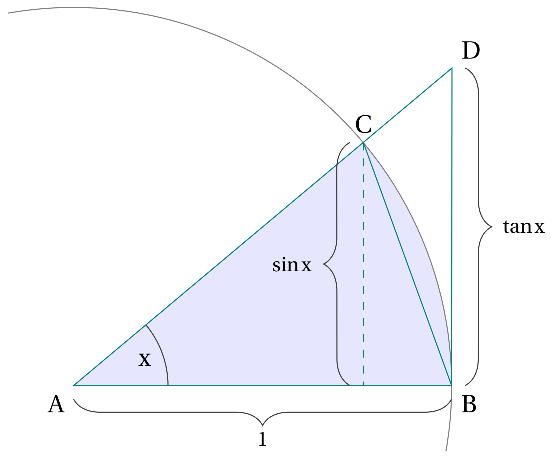
```
\tkzInterLC(Sp,F'c)(Jc,Yc)                \tkzGetSecondPoint{Fc}
\tkzInterLC(Mc,Sp)(Q,Cb)                  \tkzGetSecondPoint{A''}
\tkzDefLine[parallel=through A''](N,Mc)   \tkzGetPoint{q}

% Calculations are done, now you can draw, mark and label
\tkzDrawPolygon(A,B,C)
\tkzDrawCircle(Q,Bc)%
\tkzDrawCircle[euler,lightgray](A,B,C)
\tkzDrawCircles[ex](A,B,C B,C,A C,A,B)
\tkzDrawSegments[dashed](A,A'  C,C' A',Zc Za,C' B,Cb B,Ab A,Ca C,Ac
                         Ja,Xa Jb,Yb Jc,Zc)
\begin{scope}
   \tkzClipCircle(Q,Cb) % We limit the drawing of the lines
   \tkzDrawLine[add=5 and 12,orange](K,O)
   \tkzDrawLine[add=12 and 28,red!50!black](N,Sp)
\end{scope}
\tkzDrawPoints(A,B,C,K,Ja,Jb,Jc,Q,N,O,Sp,Mc,Xa,Xb,Yb,Yc,Za,Zc)
\tkzDrawPoints(A',C',A'',Ab,Cb,Bc,Ca,Ac,Ba,Fa,Fb,Fc,F'a,F'b,F'c)
\tkzLabelPoints(Ja,Jb,Jc,Q,Xa,Xb,Za,Zc,Ab,Cb,Bc,Ca,Ac,Ba,F'b)
\tkzLabelPoints[above](O,K,F'a,Fa,A'')
\tkzLabelPoints[below](B,F'c,Yc,N,Sp,Fc,Mc)
\tkzLabelPoints[left](A',C',Fb)
\tkzLabelPoints[right](C)
\tkzLabelPoints[below right](A)
\tkzLabelPoints[above right](Yb)
\tkzDrawSegments[color=green!50!black](Mc,N Mc,A'' A'',Q)
\tkzDrawSegments[color=red,dashed](Ac,Ab Ca,Cb Ba,Bc Ja,Jc A',Cb C',Ab)
\tkzDrawSegments[color=red](Cb,Ab Bc,Ac Ba,Ca A',C')
\tkzMarkSegments[color=red,mark=|](Cb,Ab Bc,Ac Ba,Ca)
\tkzMarkRightAngles(Jc,Zc,A Ja,Xa,B Jb,Yb,C)
\tkzDrawSegments[green,dashed](A,F'a B,F'b C,F'c)
\end{tikzpicture}

\end{document}
```

The result


F′a
Ac
A″
C′
Za
Bc
Ja
Yb
C
F′b
Jb
Fa
Fb
K
O
Q
N
Sp
A′
Ba
Xb
A
Mc
Fc
B
Xa
Ab
Yc
Zc
Jc
Ca
Cb
F′c

47. Different authors

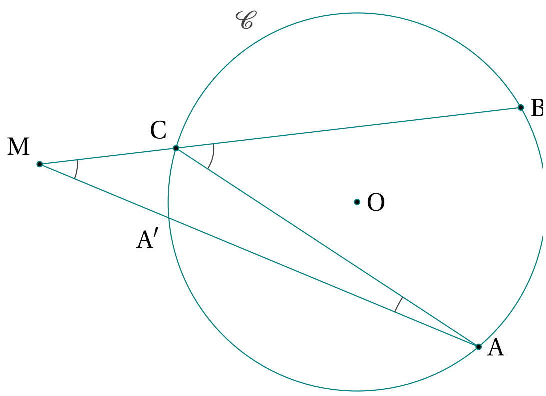
47.1. Code from Andrew Swan




```
\begin{tikzpicture}[scale=1.25]
\def\radius{4}
\def\angle{40}
\pgfmathsetmacro{\htan}{tan(\angle)}
\tkzDefPoint(0,0){A} \tkzDefPoint(0,\radius){F}
\tkzDefPoint(\radius,0){B}
\tkzDefPointBy[rotation= center A angle \angle](B)
\tkzGetPoint{C}
\tkzDefLine[perpendicular=through B,K=1](A,B)
\tkzGetPoint{b}
\tkzInterLL(A,C)(B,b) \tkzGetPoint{D}
\tkzDefLine[perpendicular=through C,K=-1](A,B)
\tkzGetPoint{c}
\tkzInterLL(C,c)(A,B) \tkzGetPoint{E}
\tkzDrawSector[fill=blue,opacity=0.1](A,B)(C)
\tkzDrawArc[thin](A,B)(F)
\tkzMarkAngle(B,A,C)
\tkzLabelAngle[pos=0.8](B,A,C){$x$}
\tkzDrawPolygon(A,B,D)
\tkzDrawSegments(C,B)
\tkzDrawSegments[dashed,thin](C,E)
\tkzLabelPoints[below left](A)
\tkzLabelPoints[below right](B)
\tkzLabelPoints[above](C)
\tkzLabelPoints[above right](D)
\begin{scope}[/pgf/decoration/raise=5pt]
\draw [decorate,decoration={brace,mirror,
   amplitude=10pt},xshift=0pt,yshift=-4pt]
(A) -- (B) node [black,midway,yshift=-20pt]
{\footnotesize $1$};
\draw [decorate,decoration={brace,amplitude=10pt},
       xshift=4pt,yshift=0pt]
(D) -- (B) node [black,midway,xshift=27pt]
{\footnotesize $\tan x$};
\draw [decorate,decoration={brace,amplitude=10pt},
       xshift=4pt,yshift=0pt]
(E) -- (C) node [black,midway,xshift=-27pt]
{\footnotesize $\sin x$};
\end{scope}
\end{tikzpicture}
```

47.2. Example: Dimitris Kapeta

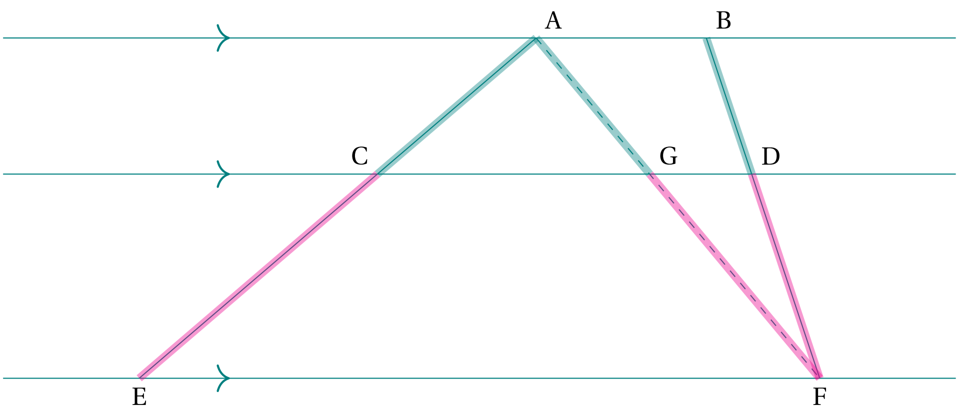
You need in this example to use `mkpos=.2` with `\tkzMarkAngle` because the measure of $\widehat{CAM}$ is too small. Another possiblity is to use `\tkzFillAngle`.




```
\begin{tikzpicture}[scale=1]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(2.5,0){N}
  \tkzDefPoint(-4.2,0.5){M}
  \tkzDefPointBy[rotation=center O angle 30](N)
  \tkzGetPoint{B}
  \tkzDefPointBy[rotation=center O angle -50](N)
  \tkzGetPoint{A}
  \tkzInterLC(M,B)(O,N) \tkzGetFirstPoint{C}
  \tkzInterLC(M,A)(O,N) \tkzGetSecondPoint{A'}
  \tkzMarkAngle[mkpos=.2, size=0.5](A,C,B)
  \tkzMarkAngle[mkpos=.2, size=0.5](A,M,C)
  \tkzDrawSegments(A,C M,A M,B)
  \tkzDrawCircle(O,N)
  \tkzLabelCircle[above left](O,N)(120){%
                 $\mathcal{C}$}
  \begin{scope}[xfp]
    \tkzMarkAngle[mkpos=.2, size=1.2](C,A,M)
  \end{scope}

  \tkzDrawPoints(O, A, B, M, B, C)
  \tkzLabelPoints[right](O,A,B)
  \tkzLabelPoints[above left](M,C)
  \tkzLabelPoint[below left](A'){$A'$}
\end{tikzpicture}
```

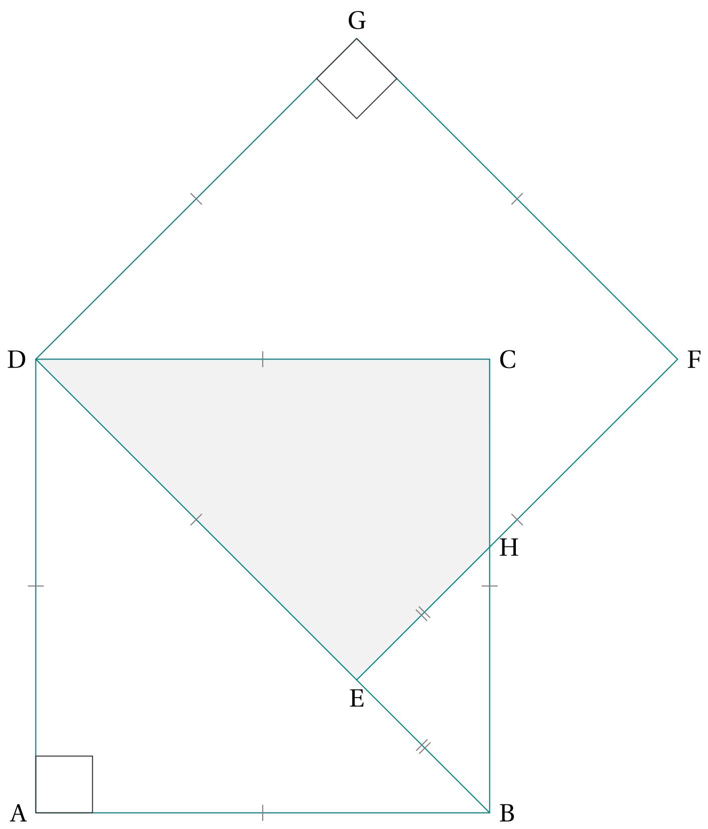
47.3. Example : John Kitzmiller

Prove that $\frac{AC}{CE} = \frac{BD}{DF}$.
Another interesting example from John, you can see how to use some extra options like **`decoration`** and **`postaction`** from Ti*k*Z with **`tkz-euclide`**.



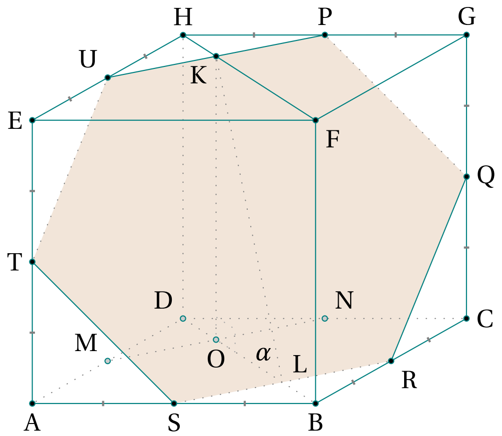
```
\begin{tikzpicture}[scale=1.5,decoration={markings,
  mark=at position 3cm with {\arrow[scale=2]{>}}}]
  \tkzDefPoints{0/0/E, 6/0/F, 0/1.8/P, 6/1.8/Q, 0/3/R, 6/3/S}
  \tkzDrawLines[postaction={decorate}](E,F P,Q R,S)
  \tkzDefPoints{3.5/3/A, 5/3/B}
  \tkzDrawSegments(E,A F,B)
  \tkzInterLL(E,A)(P,Q) \tkzGetPoint{C}
  \tkzInterLL(B,F)(P,Q) \tkzGetPoint{D}
  \tkzLabelPoints[above right](A,B)
  \tkzLabelPoints[below](E,F)
  \tkzLabelPoints[above left](C)
  \tkzDrawSegments[style=dashed](A,F)
  \tkzInterLL(A,F)(P,Q) \tkzGetPoint{G}
  \tkzLabelPoints[above right](D,G)
  \tkzDrawSegments[color=teal, line width=3pt, opacity=0.4](A,C A,G)
  \tkzDrawSegments[color=magenta, line width=3pt, opacity=0.4](C,E G,F)
  \tkzDrawSegments[color=teal, line width=3pt, opacity=0.4](B,D)
  \tkzDrawSegments[color=magenta, line width=3pt, opacity=0.4](D,F)
\end{tikzpicture}
```

47.4. Example 1: from Indonesia



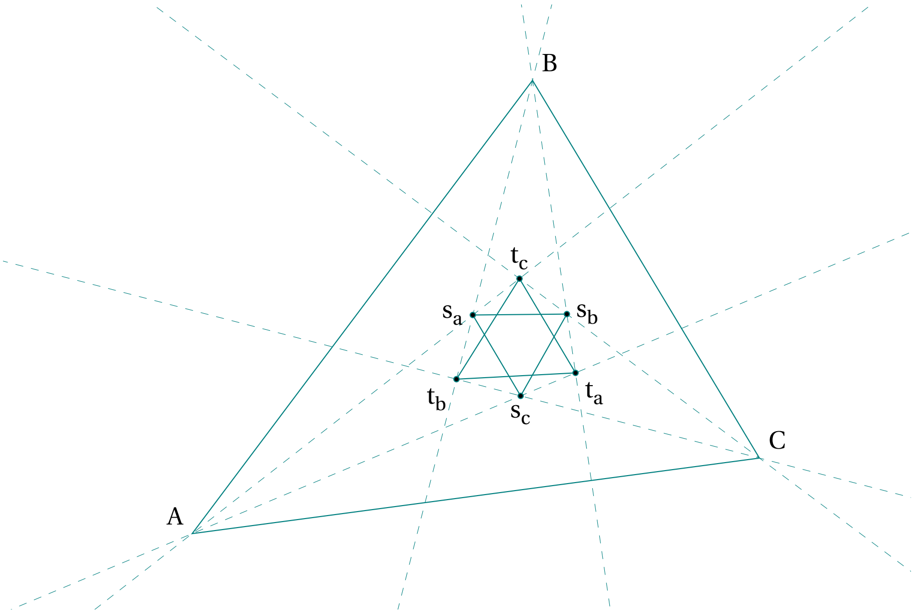
```
\begin{tikzpicture}[scale=3]
   \tkzDefPoints{0/0/A,2/0/B}
   \tkzDefSquare(A,B) \tkzGetPoints{C}{D}
   \tkzDefPointBy[rotation=center D angle 45](C)\tkzGetPoint{G}
   \tkzDefSquare(G,D)\tkzGetPoints{E}{F}
   \tkzInterLL(B,C)(E,F)\tkzGetPoint{H}
   \tkzFillPolygon[gray!10](D,E,H,C,D)
   \tkzDrawPolygon(A,...,D)\tkzDrawPolygon(D,...,G)
   \tkzDrawSegment(B,E)
   \tkzMarkSegments[mark=|,size=3pt,color=gray](A,B B,C C,D D,A E,F F,G G,D D,E)
   \tkzMarkSegments[mark=||,size=3pt,color=gray](B,E E,H)
   \tkzLabelPoints[left](A,D)
   \tkzLabelPoints[right](B,C,F,H)
   \tkzLabelPoints[above](G)\tkzLabelPoints[below](E)
   \tkzMarkRightAngles(D,A,B D,G,F)
\end{tikzpicture}
```

47.5. Example 2: from Indonesia



```
  \begin{tikzpicture}[pol/.style={fill=brown!40,opacity=.5},
      seg/.style={tkzdotted,color=gray}, hidden pt/.style={fill=gray!40},
       mra/.style={color=gray!70,tkzdotted,/tkzrightangle/size=.2},scale=1.5]
  \tkzDefPoints{0/0/A,2.5/0/B,1.33/0.75/D,0/2.5/E,2.5/2.5/F}
  \tkzDefLine[parallel=through D](A,B)  \tkzGetPoint{I1}
  \tkzDefLine[parallel=through B](A,D)  \tkzGetPoint{I2}
  \tkzInterLL(D,I1)(B,I2)               \tkzGetPoint{C}
  \tkzDefLine[parallel=through E](A,D)  \tkzGetPoint{I3}
  \tkzDefLine[parallel=through D](A,E)  \tkzGetPoint{I4}
  \tkzInterLL(E,I3)(D,I4)               \tkzGetPoint{H}
  \tkzDefLine[parallel=through F](E,H)  \tkzGetPoint{I5}
  \tkzDefLine[parallel=through H](E,F)  \tkzGetPoint{I6}
  \tkzInterLL(F,I5)(H,I6)               \tkzGetPoint{G}
  \tkzDefMidPoint(G,H) \tkzGetPoint{P}  \tkzDefMidPoint(G,C) \tkzGetPoint{Q}
  \tkzDefMidPoint(B,C) \tkzGetPoint{R}  \tkzDefMidPoint(A,B) \tkzGetPoint{S}
  \tkzDefMidPoint(A,E) \tkzGetPoint{T}  \tkzDefMidPoint(E,H) \tkzGetPoint{U}
  \tkzDefMidPoint(A,D) \tkzGetPoint{M}  \tkzDefMidPoint(D,C) \tkzGetPoint{N}
  \tkzInterLL(B,D)(S,R)\tkzGetPoint{L} \tkzInterLL(H,F)(U,P) \tkzGetPoint{K}
  \tkzDefLine[parallel=through K](D,H)  \tkzGetPoint{I7}
  \tkzInterLL(K,I7)(B,D)                \tkzGetPoint{O}
  \tkzFillPolygon[pol](P,Q,R,S,T,U)
  \tkzDrawSegments[seg](K,O K,L P,Q R,S T,U C,D H,D A,D M,N B,D)
  \tkzDrawSegments(E,H B,C G,F G,H G,C Q,R S,T U,P H,F)
  \tkzDrawPolygon(A,B,F,E)
  \tkzDrawPoints(A,B,C,E,F,G,H,P,Q,R,S,T,U,K) \tkzDrawPoints[hidden pt](M,N,O,D)
  \tkzMarkRightAngle[mra](L,O,K)
  \tkzMarkSegments[mark=|,size=1pt,thick,color=gray](A,S B,S B,R C,R
                   Q,C Q,G G,P H,P E,U H,U E,T A,T)
  \tkzLabelAngle[pos=.3](K,L,O){$\alpha$}
  \tkzLabelPoints[below](O,A,S,B)    \tkzLabelPoints[above](H,P,G)
  \tkzLabelPoints[left](T,E)         \tkzLabelPoints[right](C,Q)
  \tkzLabelPoints[above left](U,D,M) \tkzLabelPoints[above right](L,N)
  \tkzLabelPoints[below right](F,R)  \tkzLabelPoints[below left](K)
\end{tikzpicture}
```

47.6. Illustration of the Morley theorem by Nicolas François

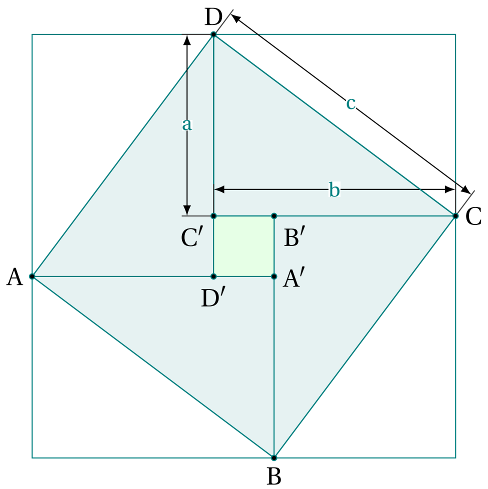



```
\begin{tikzpicture}
  \tkzInit[ymin=-3,ymax=5,xmin=-5,xmax=7]
  \tkzClip
  \tkzDefPoints{-2.5/-2/A,2/4/B,5/-1/C}
  \tkzFindAngle(C,A,B) \tkzGetAngle{anglea}
  \tkzDefPointBy[rotation=center A angle 1*\anglea/3](C) \tkzGetPoint{TA1}
  \tkzDefPointBy[rotation=center A angle 2*\anglea/3](C) \tkzGetPoint{TA2}
  \tkzFindAngle(A,B,C) \tkzGetAngle{angleb}
  \tkzDefPointBy[rotation=center B angle 1*\angleb/3](A) \tkzGetPoint{TB1}
  \tkzDefPointBy[rotation=center B angle 2*\angleb/3](A) \tkzGetPoint{TB2}
  \tkzFindAngle(B,C,A) \tkzGetAngle{anglec}
  \tkzDefPointBy[rotation=center C angle 1*\anglec/3](B) \tkzGetPoint{TC1}
  \tkzDefPointBy[rotation=center C angle 2*\anglec/3](B) \tkzGetPoint{TC2}
  \tkzInterLL(A,TA1)(B,TB2) \tkzGetPoint{U1}
  \tkzInterLL(A,TA2)(B,TB1) \tkzGetPoint{V1}
  \tkzInterLL(B,TB1)(C,TC2) \tkzGetPoint{U2}
  \tkzInterLL(B,TB2)(C,TC1) \tkzGetPoint{V2}
  \tkzInterLL(C,TC1)(A,TA2) \tkzGetPoint{U3}
  \tkzInterLL(C,TC2)(A,TA1) \tkzGetPoint{V3}
  \tkzDrawPolygons(A,B,C U1,U2,U3 V1,V2,V3)
  \tkzDrawLines[add=2 and 2,very thin,dashed](A,TA1 B,TB1 C,TC1 A,TA2 B,TB2 C,TC2)
  \tkzDrawPoints(U1,U2,U3,V1,V2,V3)
  \tkzLabelPoint[left](V1){$s_a$} \tkzLabelPoint[right](V2){$s_b$}
  \tkzLabelPoint[below](V3){$s_c$} \tkzLabelPoint[above left](A){$A$}
  \tkzLabelPoints[above right](B,C) \tkzLabelPoint(U1){$t_a$}
  \tkzLabelPoint[below left](U2){$t_b$} \tkzLabelPoint[above](U3){$t_c$}
\end{tikzpicture}
```

47.7. Gou gu theorem / Pythagorean Theorem by Zhao Shuang

Gou gu theorem / Pythagorean Theorem by Zhao Shuang

Pythagoras was not the first person who discovered this theorem around the world. Ancient China discovered this theorem much earlier than him. So there is another name for the Pythagorean theorem in China, the Gou-Gu theorem. Zhao Shuang was an ancient Chinese mathematician. He rediscovered the "Gou gu therorem", which is actually the Chinese version of the "Pythagorean theorem". Zhao Shuang used a method called the "cutting and compensation principle", he created a picture of "Pythagorean Round Square" Below the figure used to illustrate the proof of the "Gou gu theorem." (code from Nan Geng)

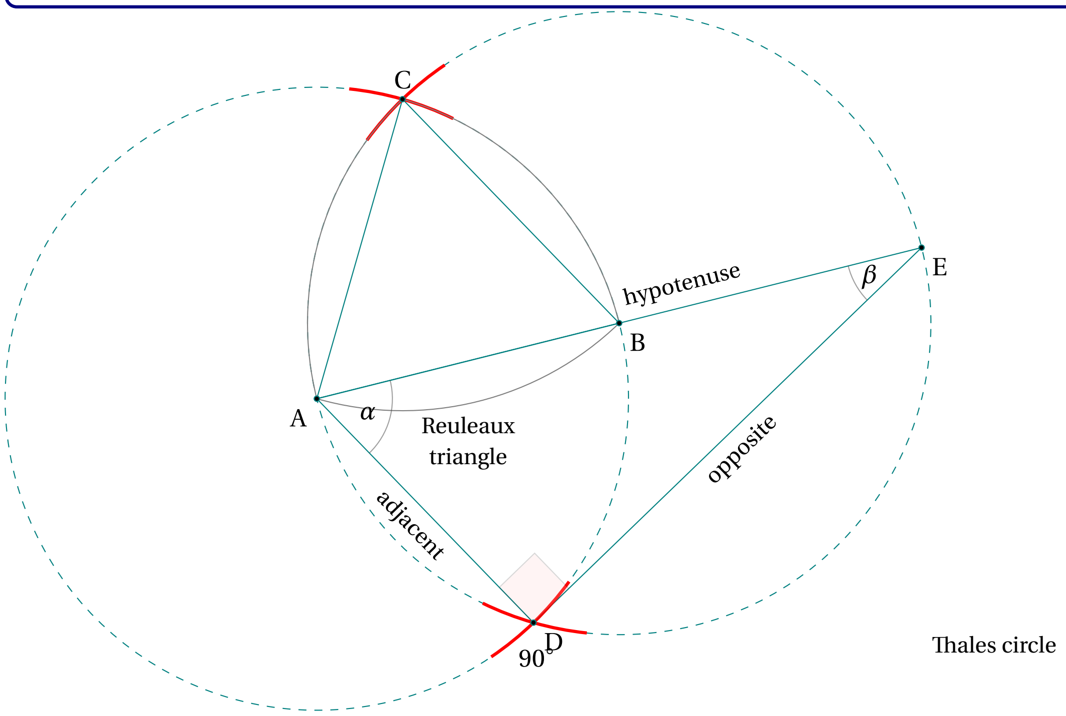



```
\begin{tikzpicture}[scale=.8]
  \tkzDefPoint(0,0){A} \tkzDefPoint(4,0){A'}
  \tkzInterCC[R](A, 5)(A', 3)
  \tkzGetSecondPoint{B}
  \tkzDefSquare(A,B)   \tkzGetPoints{C}{D}
  \tkzCalcLength(A,A') \tkzGetLength{lA}
  \tkzCalcLength(A',B) \tkzGetLength{lB}
  \pgfmathparse{\lA-\lB}
  \tkzInterLC[R](A,A')(A',\pgfmathresult)
  \tkzGetFirstPoint{D'}
  \tkzDefSquare(D',A')\tkzGetPoints{B'}{C'}
  \tkzDefLine[orthogonal=through D](D,D')
   \tkzGetPoint{d}
  \tkzDefLine[orthogonal=through A](A,A')
   \tkzGetPoint{a}
  \tkzDefLine[orthogonal=through C](C,C')
   \tkzGetPoint{c}
  \tkzInterLL(D,d)(C,c) \tkzGetPoint{E}
  \tkzInterLL(D,d)(A,a) \tkzGetPoint{F}
  \tkzDefSquare(E,F)\tkzGetPoints{G}{H}
  \tkzDrawPolygons[fill=teal!10](A,B,A' B,C,B'
     C,D,C' A,D',D)
  \tkzDrawPolygons(A,B,C,D E,F,G,H)
  \tkzDrawPolygon[fill=green!10](A',B',C',D')
  \tkzDrawSegment[dim={$a$,-10pt,}](D,C')
  \tkzDrawSegment[dim={$b$,-10pt,}](C,C')
  \tkzDrawSegment[dim={$c$,-10pt,}](C,D)
  \tkzDrawPoints[size=2](A,B,C,D,A',B',C',D')
  \tkzLabelPoints[left](A)
  \tkzLabelPoints[below](B)
  \tkzLabelPoints[right](C)
  \tkzLabelPoints[above](D)
  \tkzLabelPoints[right](A')
  \tkzLabelPoints[below right](B')
  \tkzLabelPoints[below left](C')
  \tkzLabelPoints[below](D')
 \end{tikzpicture}
```

47.8. Reuleaux-Triangle

Reuleaux-triangle by Stefan Kottwitz

A well-known classic field of mathematics is geometry. You may know Euclidean geometry from school, with constructions by compass and ruler. Math teachers may be very interested in drawing geometry constructions and explanations. Underlying constructions can help us with general drawings where we would need intersections and tangents of lines and circles, even if it does not look like geometry. So, here, we will remember school geometry drawings. We will use the tkz-euclide package, which works on top of TikZ. We will construct an equilateral triangle. Then we extend it to get a Reuleaux triangle, and add annotations. The code is fully explained in the LaTeX Cookbook, Chapter 10, Advanced Mathematics, Drawing geometry pictures. Stefan Kottwitz



Part X.

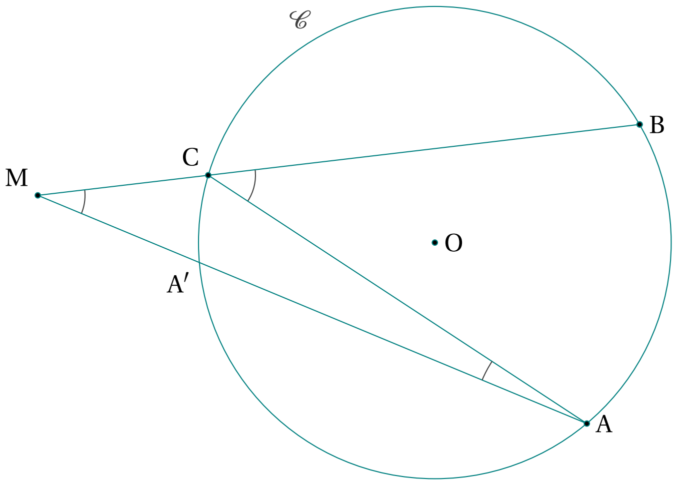
FAQ

48. FAQ

48.1. Most common errors

For the moment, I'm basing myself on my own, because having changed syntax several times, I've made a number of mistakes. This section is going to be expanded.

– Error "dimension too large" : In some cases, this error occurs. One way to avoid it is to use the "**xfp**" option. When this option is used in an environment, the "veclen" function is replaced by a function dependent on "xfp". For example, an error occurs if you use the macro `\tkzDrawArc` with too small an angle. The error is produced by the `decoration` library when you want to place a mark on an arc. Even if the mark is absent, the error is still present.




```
\begin{tikzpicture}[scale=1.25]
  \tkzDefPoint(0,0){O}
  \tkzDefPoint(2.5,0){N}
  \tkzDefPoint(-4.2,0.5){M}
  \tkzDefPointBy[rotation=center O angle 30](N)
  \tkzGetPoint{B}
  \tkzDefPointBy[rotation=center O angle -50](N)
  \tkzGetPoint{A}
  \tkzInterLC(M,B)(O,N) \tkzGetFirstPoint{C}
  \tkzInterLC(M,A)(O,N) \tkzGetSecondPoint{A'}
  \tkzMarkAngle[mkpos=.2, size=0.5](A,C,B)
  \tkzMarkAngle[mkpos=.2, size=0.5](A,M,C)
  \tkzDrawSegments(A,C M,A M,B)
  \tkzDrawCircle(O,N)
  \tkzLabelCircle[above left](O,N)(120){$\mathcal{C}$}
  \begin{scope}[xfp]
    \tkzMarkAngle[mkpos=.2, size=1.2](C,A,M)
  \end{scope}
  \tkzDrawPoints(O, A, B, M, B, C)
  \tkzLabelPoints[right](O,A,B)
  \tkzLabelPoints[above left](M,C)
  \tkzLabelPoint[below left](A'){$A'$}
\end{tikzpicture}
```

– `\tkzDrawPoint(A,B)` when you need `\tkzDrawPoints`.

– `\tkzGetPoint(A)` When defining an object, use braces and not brackets, so write: `\tkzGetPoint{A}`.

- `\tkzGetPoint{A}` in place of `\tkzGetFirstPoint{A}`. When a macro gives two points as results, either we retrieve these points using `\tkzGetPoints{A}{B}`, or we retrieve only one of the two points, using `\tkzGetFirstPoint{A}` or `\tkzGetSecondPoint{A}`. These two points can be used with the reference `tkzFirstPointResult` or `tkzSecondPointResult`. It is possible that a third point is given as `tkzPointResult`.

- `\tkzDrawSegment(A,B A,C)` when you need `\tkzDrawSegments`. It is possible to use only the versions with an "s" but it is less efficient!

- Mixing options and arguments; all macros that use a circle need to know the radius of the circle. If the radius is given by a measure then the option includes a `R`.

- `\tkzDrawSegments[color = gray,style=dashed]{B,B' C,C'}` is a mistake. Only macros that define an object use braces.

- The angles are given in degrees, more rarely in radians.

- If an error occurs in a calculation when passing parameters, then it is better to make these calculations before calling the macro.

- Do not mix the syntax of `pgfmath` and `xfp`. I've often chosen `xfp` but if you prefer pgfmath then do your calculations before passing parameters.

- Use of `\tkzClip`: In order to get accurate results, I avoided using normalized vectors. The advantage of normalization is to control the dimension of the manipulated objects, the disadvantage is that with TeX, this implies inaccuracies. These inaccuracies are often small, in the order of a thousandth, but they lead to disasters if the drawing is enlarged. Not normalizing implies that some points are far away from the working area and `\tkzClip` allows you to reduce the size of the drawing.

Index

\add, 109
\ang, 98
\Ax, 167
\Ay, 167

\coordinate, 23

\dAB, 165
\Delta, 152
\draw (A)--(B);, 110

\endpgfinterruptboundingbox, 138
Environment
    scope, 25, 170
    tikzpicture, 170
    tikzspicture, 170

\fpeval, 93

\len, 166, 167

Operating System
    Windows, 5

Package
    fp, 20
    pgfmath, 217
    tkz-euclide, 132
    xfp, 5, 20, 23, 25, 92, 165, 217
\pgfinterruptboundingbox, 138
\pgflinewidth, 107, 108
\pgfmathsetmacro, 93
\pgfresetboundingbox, 132
\px, 167
\py, 167

\rAB, 30

standalone, 10

TeX Distributions
    MiKTeX, 5
    TeXLive, 5
Ti*k*Z Library
    angles, 20
    babel, 15
    decoration, 216
    quotes, 20
\tikzset, 169
\tkzAngleResult, 98, 100
\tkzAutoLabelPoints, 149
\tkzCalcLength, 165
\tkzCalcLength: arguments
    (pt1,pt2){name of macro}, 165
\tkzCalcLength: options
    cm, 165
\tkzCalcLength[⟨local options⟩](⟨pt1,pt2⟩), 165
\tkzCentroid, 31

\tkzClip, 15, 20, 132, 133, 217
\tkzClip: options
    space, 133
\tkzClipBB, 20, 134
\tkzClipCircle[out], 139
\tkzClipCircle, 79, 136
\tkzClipCircle: arguments
    (⟨A,B⟩) or (⟨A,r⟩), 136
\tkzClipCircle: options
    R, 136
    out, 136
    radius, 136
\tkzClipCircle[⟨local options⟩](⟨A,B⟩) or (⟨A,r⟩), 136
\tkzClipPolygon[out], 135, 139
\tkzClipPolygon, 135
\tkzClipPolygon: arguments
    (⟨pt1,pt2,pt3,…⟩), 135
\tkzClipPolygon: options
    out, 135
\tkzClipPolygon[⟨local options⟩](⟨points list⟩), 135
\tkzClipSector(O,A)(B), 137
\tkzClipSector[R](O,2)(30,90), 137
\tkzClipSector[rotate](O,A)(90), 137
\tkzClipSector, 137
\tkzClipSector: options
    R, 137
    rotate, 137
    towards, 137
\tkzClipSector[⟨local options⟩](⟨O,…⟩)(⟨…⟩), 137
\tkzClip[⟨local options⟩], 133
\tkzcmtopt, 167
\tkzcmtopt: arguments
    (nombre){name of macro}, 167
\tkzcmtopt(⟨nombre⟩){⟨name of macro⟩}, 167
\tkzCompass, 121, 158
\tkzCompass: options
    delta, 158
    length, 158
\tkzCompasss, 158
\tkzCompasss: options
    delta, 158
    length, 158
\tkzCompasss[⟨local options⟩](⟨pt1,pt2 pt3,pt4,…⟩), 158
\tkzCompass[⟨local options⟩](⟨A,B⟩), 158
\tkzDefBarycentricPoint, 31, 33
\tkzDefBarycentricPoint: arguments
    (pt1=α_1,pt2=α_2,…), 31
\tkzDefBarycentricPoint(⟨pt1=α_1,pt2=α_2,…⟩), 31
\tkzDefCircle[radius](A,B), 166
\tkzDefCircle, 79
\tkzDefCircle: arguments
    (⟨pt1,pt2⟩) or (⟨pt1,pt2,pt3⟩), 79
\tkzDefCircle: options
    K, 79
    apollonius, 79
    circum, 79
    diameter, 79
    euler or nine, 79

ex, 79
in, 79
spieker, 79
through, 79
\tkzDefCircleBy, 85
\tkzDefCircleBy: arguments
pt1,pt2, 85
\tkzDefCircleBy: options
homothety, 85
inversion, 85
orthogonal from, 85
orthogonal through, 85
projection , 85
reflection, 85
rotation , 85
symmetry , 85
translation, 85
\tkzDefCircleBy[⟨local options⟩](⟨pt1,pt2⟩), 85
\tkzDefCirclesBy, 85
\tkzDefCircle[⟨local options⟩](⟨A,B⟩) or (⟨A,B,C⟩), 79
\tkzDefEquiPoints, 33
\tkzDefEquiPoints: arguments
(pt1,pt2), 33
\tkzDefEquiPoints: options
/compass/delta, 33
dist, 33
from=pt, 33
show, 33
\tkzDefEquiPoints[⟨local options⟩](⟨pt1,pt2⟩), 33
\tkzDefGoldenRatio(A,C), 32
\tkzDefGoldenRatio, 32
\tkzDefGoldenRatio: arguments
(pt1,pt2), 32
\tkzDefGoldenRatio(⟨pt1,pt2⟩), 32
\tkzDefGoldenRectangle, 76
\tkzDefGoldenRectangle: arguments
(⟨pt1,pt2⟩), 76
\tkzDefGoldenRectangle(⟨point,point⟩), 76
\tkzDefGoldRectangle, 76
\tkzDefLine, 58
\tkzDefLine: arguments
(⟨pt1,pt2,pt3⟩), 58
(⟨pt1,pt2⟩), 58
\tkzDefLine: options
K, 58
bisector out, 58
bisector, 58
mediator, 58
normed, 58
orthogonal=through..., 58
parallel=through..., 58
perpendicular=through..., 58
\tkzDefLine[⟨local options⟩](⟨pt1,pt2⟩) or (⟨pt1,pt2,pt3⟩), 58
\tkzDefMidPoint, 8, 30
\tkzDefMidPoint: arguments
(pt1,pt2), 30
\tkzDefMidPoint(⟨pt1,pt2⟩), 30
\tkzDefParallelogram, 75

\tkzDefParallelogram: arguments
(⟨pt1,pt2,pt3⟩), 75
\tkzDefParallelogram(⟨pt1,pt2,pt3⟩), 75
\tkzDefPoint, 23, 24, 30, 89
\tkzDefPoint: arguments
(α:d), 24
(x,y), 24
{ref}, 24
\tkzDefPoint: options
label, 24
shift, 24
\tkzDefPointBy, 43
\tkzDefPointBy: arguments
pt, 43
\tkzDefPointBy: options
homothety, 43
inversion negative, 43
inversion, 43
projection , 43
reflection, 43
rotation in rad, 43
rotation , 43
symmetry , 43
translation, 43
\tkzDefPointBy[⟨local options⟩](⟨pt⟩), 43
\tkzDefPointOnCircle, 42
\tkzDefPointOnCircle: options
angle, 42
center, 42
radius, 42
\tkzDefPointOnCircle[⟨local options⟩], 42
\tkzDefPointOnLine, 41
\tkzDefPointOnLine: arguments
pt1,pt2, 41
\tkzDefPointOnLine: options
pos=nb, 41
\tkzDefPointOnLine[⟨local options⟩](⟨A,B⟩), 41
\tkzDefPoints{0/0/O,2/2/A}, 27
\tkzDefPoints, 23, 27
\tkzDefPoints: arguments
$x_i/y_i/r_i$, 27
\tkzDefPoints: options
shift, 27
\tkzDefPointsBy, 43, 50
\tkzDefPointsBy: arguments
(⟨list of points⟩){⟨list of pts⟩}, 50
\tkzDefPointsBy: options
homothety = center #1 ratio #2, 50
projection = onto #1--#2, 50
reflection = over #1--#2, 50
rotation = center #1 angle #2, 50
rotation in rad = center #1 angle #2, 50
symmetry = center #1, 50
translation = from #1 to #2, 50
\tkzDefPointsBy[⟨local options⟩](⟨list of points⟩){⟨list of points⟩}, 50
\tkzDefPoints[⟨local options⟩]{⟨$x_1/y_1/n_1,x_2/y_2/r_2$, ...⟩}, 27
\tkzDefPointWith, 52
\tkzDefPointWith: arguments

(pt1,pt2), 52
\tkzDefPointWith: options
K, 52
colinear normed= at #1, 52
colinear= at #1, 52
linear normed, 52
linear, 52
orthogonal normed, 52
orthogonal, 52
\tkzDefPointWith(⟨pt1,pt2⟩), 52
\tkzDefPoint[⟨local options⟩](⟨x,y⟩){⟨ref⟩} or (⟨α:d⟩){⟨ref⟩}, 24
\tkzDefProjExcenter[name=J](A,B,C)(a,b,c){Y,Z,X}, 40
\tkzDefProjExcenter, 40
\tkzDefProjExcenter: arguments
(pt1=α_1,pt2=α_2,...), 40
\tkzDefProjExcenter: options
name, 40
\tkzDefProjExcenter[⟨local options⟩](⟨A,B,C⟩)(⟨a,b,c⟩){⟨X,Y,Z⟩}, 40
\tkzDefRandPointOn, 21, 102
\tkzDefRandPointOn: options
circle =center pt1 radius dim, 102
circle through=center pt1 through pt2, 102
disk through=center pt1 through pt2, 102
line=pt1--pt2, 102
rectangle=pt1 and pt2, 102
segment= pt1--pt2, 102
\tkzDefRandPointOn[⟨local options⟩], 102
\tkzDefRectangle, 75
\tkzDefRectangle: arguments
(⟨pt1,pt2⟩), 75
\tkzDefRectangle(⟨pt1,pt2⟩), 75
\tkzDefRegPolygon, 77
\tkzDefRegPolygon: arguments
(⟨pt1,pt2⟩), 77
\tkzDefRegPolygon: options
Options TikZ, 77
center, 77
name, 77
sides, 77
side, 77
\tkzDefRegPolygon[⟨local options⟩](⟨pt1,pt2⟩), 77
\tkzDefShiftPoint, 25, 26
\tkzDefShiftPoint: arguments
(α:d), 26
(x,y), 26
{ref}, 26
\tkzDefShiftPoint: options
[pt], 26
\tkzDefShiftPoint[⟨Point⟩](⟨x,y⟩){⟨ref⟩} or (⟨α:d⟩){⟨ref⟩}, 26
\tkzDefSpcTriangle[medial,name=M_](A,B,C){A,B,C}, 66
\tkzDefSpcTriangle[medial,name=M](A,B,C){_A,_B,_C}, 66
\tkzDefSpcTriangle[medial](A,B,C){a,b,c}, 66
\tkzDefSpcTriangle, 66
\tkzDefSpcTriangle: options
centroid or medial, 66
euler, 66
ex or excentral, 66
extouch, 66

feuerbach, 66
in or incentral, 66
intouch or contact, 66
name, 66
orthic, 66
symmedial, 66
tangential, 66
, 66
\tkzDefSpcTriangle[⟨local options⟩](⟨p1,p2,p3⟩){⟨r1,r2,r3⟩}, 66
\tkzDefSquare, 74, 75
\tkzDefSquare: arguments
(⟨pt1,pt2⟩), 74
\tkzDefSquare(⟨pt1,pt2⟩), 74
\tkzDefTangent, 60
\tkzDefTangent: arguments
(⟨pt1,pt2 or (⟨pt1,dim⟩)⟩) , 61
\tkzDefTangent: options
at=pt, 61
from with R=pt, 61
from=pt, 61
\tkzDefTangent[⟨local options⟩](⟨pt1,pt2⟩) or (⟨pt1,dim⟩), 60
\tkzDefTriangle, 21, 63
\tkzDefTriangle: options
cheops, 63
equilateral, 63
euclid, 63
golden, 63
gold, 63
isosceles right, 63
pythagore, 63
school, 63
two angles= #1 and #2, 63
\tkzDefTriangleCenter[ortho](B,C,A), 34
\tkzDefTriangleCenter, 34
\tkzDefTriangleCenter: arguments
(pt1,pt2,pt3), 34
\tkzDefTriangleCenter: options
centroid, 34
circum, 34
euler, 34
ex, 34
feuerbach, 34
gergonne, 34
grebe, 34
in, 34
lemoine, 34
median, 34
mittenpunkt, 34
nagel, 34
orthic, 34
ortho, 34
spieker, 34
symmedian, 34
\tkzDefTriangleCenter[⟨local options⟩](⟨A,B,C⟩), 34
\tkzDefTriangle[⟨local options⟩](⟨A,B⟩), 63
\tkzDrawArc[angles](O,A)(0,90), 120
\tkzDrawArc[delta=10](O,A)(B), 120
\tkzDrawArc[R with nodes](O,2)(A,B), 120

\tkzDrawArc[R](O,2)(30,90), 120
\tkzDrawArc[rotate,color=red](O,A)(90), 120
\tkzDrawArc, 120, 216
\tkzDrawArc: options
    R with nodes, 120
    R, 120
    angles, 120
    delta, 120
    rotate, 120
    towards, 120
\tkzDrawArc[⟨local options⟩](⟨O,...⟩)(⟨...⟩), 120
\tkzDrawCircle, 79, 115, 126
\tkzDrawCircle: arguments
    (⟨pt1,pt2⟩), 115
\tkzDrawCircle: options
    R, 115
    diameter, 115
    through, 115
\tkzDrawCircles, 116
\tkzDrawCircles: arguments
    (⟨pt1,pt2 pt3,pt4 ...⟩), 116
\tkzDrawCircles: options
    R, 116
    diameter, 116
    through, 116
\tkzDrawCircles[⟨local options⟩](⟨A,B C,D ...⟩), 116
\tkzDrawCircle[⟨local options⟩](⟨A,B⟩), 115
\tkzDrawLine, 109
\tkzDrawLine: options
    TikZ options, 109
    ..., 109
    add, 109
\tkzDrawLines, 109
\tkzDrawLines[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩), 109
\tkzDrawLine[⟨local options⟩](⟨pt1,pt2⟩) , 109
\tkzDrawPoint(A,B), 216
\tkzDrawPoint, 107
\tkzDrawPoint: arguments
    name of point, 107
\tkzDrawPoint: options
    TikZ options, 107
    color, 107
    shape, 107
    size, 107
\tkzDrawPoints(A,B,C), 108
\tkzDrawPoints, 107, 108, 216
\tkzDrawPoints: arguments
    points list, 108
\tkzDrawPoints: options
    color, 108
    shape, 108
    size, 108
\tkzDrawPoints[⟨local options⟩](⟨liste⟩), 108
\tkzDrawPoint[⟨local options⟩](⟨name⟩), 107
\tkzDrawPolygon, 113
\tkzDrawPolygon: arguments
    (⟨pt1,pt2,pt3,...⟩), 113
\tkzDrawPolygon: options

Options TikZ, 113
\tkzDrawPolygons, 21
\tkzDrawPolygon[⟨local options⟩](⟨points list⟩), 113
\tkzDrawPolySeg, 114
\tkzDrawPolySeg: arguments
(⟨pt1,pt2,pt3,...⟩), 114
\tkzDrawPolySeg: options
Options TikZ, 114
\tkzDrawPolySeg[⟨local options⟩](⟨points list⟩), 114
\tkzDrawSector(O,A)(B), 123
\tkzDrawSector[R with nodes](O,2)(A,B), 123
\tkzDrawSector[R,color=teal](O,2)(30,90), 123
\tkzDrawSector[rotate,color=red](O,A)(90), 123
\tkzDrawSector, 123–125
\tkzDrawSector: options
R with nodes, 123
R, 123
rotate, 123
towards, 123
\tkzDrawSector[⟨local options⟩](⟨O,…⟩)(⟨…⟩), 123
\tkzDrawSegment(A,B A,C), 217
\tkzDrawSegment, 20, 110
\tkzDrawSegment: arguments
(pt1,pt2), 110
\tkzDrawSegment: options
TikZ options, 110
…, 110
dim, 110
\tkzDrawSegments[color = gray,style=dashed]{B,B' C,C'}, 217
\tkzDrawSegments, 112, 217
\tkzDrawSegments[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩), 112
\tkzDrawSegment[⟨local options⟩](⟨pt1,pt2⟩), 110
\tkzDrawSemiCircle, 119
\tkzDrawSemiCircle: arguments
(⟨pt1,pt2⟩), 119
\tkzDrawSemiCircle: options
diameter, 119
through, 119
\tkzDrawSemiCircles, 21, 119
\tkzDrawSemiCircles: arguments
(⟨pt1,pt2 pt3,pt4 ...⟩), 119
\tkzDrawSemiCircles: options
diameter, 119
through, 119
\tkzDrawSemiCircles[⟨local options⟩](⟨A,B C,D …⟩), 119
\tkzDrawSemiCircle[⟨local options⟩](⟨A,B⟩), 119
\tkzDrawTriangles, 21
\tkzDuplicateLen, 164
\tkzDuplicateLength, 164
\tkzDuplicateSegment, 164
\tkzDuplicateSegment: arguments
(pt1,pt2)(pt3,pt4){pt5}, 164
\tkzDuplicateSegment(⟨pt1,pt2⟩)(⟨pt3,pt4⟩){⟨pt5⟩}, 164
\tkzFillAngle, 129, 130, 207
\tkzFillAngle: options
size, 130
\tkzFillAngles, 130
\tkzFillAngles[⟨local options⟩](⟨A,O,B⟩)(⟨A',O',B'⟩)etc., 130

\tkzFillAngle[⟨local options⟩](⟨A,O,B⟩), 130
\tkzFillCircle, 79, 126
\tkzFillCircle: options
    R, 126
    radius, 126
\tkzFillCircle[⟨local options⟩](⟨A,B⟩), 126
\tkzFillPolygon, 128
\tkzFillPolygon: arguments
    (⟨pt1,pt2,…⟩), 128
\tkzFillPolygon[⟨local options⟩](⟨points list⟩), 128
\tkzFillSector(O,A)(B), 129
\tkzFillSector[R with nodes](O,2)(A,B), 129
\tkzFillSector[R,color=blue](O,2)(30,90), 129
\tkzFillSector[rotate,color=red](O,A)(90), 129
\tkzFillSector, 128, 129
\tkzFillSector: options
    R with nodes, 129
    R, 129
    rotate, 129
    towards, 129
\tkzFillSector[⟨local options⟩](⟨O,…⟩)(⟨…⟩), 129
\tkzFindAngle, 98
\tkzFindAngle: arguments
    (pt1,pt2,pt3), 98
\tkzFindAngle(⟨pt1,pt2,pt3⟩), 98
\tkzFindSlopeAngle, 100, 101
\tkzFindSlopeAngle: arguments
    (pt1,pt2), 100
\tkzFindSlopeAngle(⟨A,B⟩), 100
\tkzGetAngle, 98, 100
\tkzGetAngle: arguments
    name of macro, 98
\tkzGetAngle(⟨name of macro⟩), 98
\tkzGetFirstPoint{A}, 217
\tkzGetFirstPoint{Jb}, 82
\tkzGetFirstPoint{M} , 29
\tkzGetFirstPoint, 29, 74
\tkzGetFirstPoint: arguments
    ref1, 29
\tkzGetFirstPoint{⟨ref1⟩}, 29
\tkzGetLength{dAB}, 165
\tkzGetLength, 30, 79, 165, 166
\tkzGetLength: arguments
    name of a macro, 30
\tkzGetLength{⟨name of a macro⟩}, 30
\tkzGetPoint(A), 216
\tkzGetPoint{A}, 216, 217
\tkzGetPoint{C}, 52
\tkzGetPoint{M} , 29
\tkzGetPoint{M}, 43
\tkzGetPoint, 8, 21, 29, 30, 32, 34–36, 52, 58, 63, 76, 79, 102
\tkzGetPoint: arguments
    ref, 29
\tkzGetPointCoord, 167
\tkzGetPointCoord: arguments
    (point){name of macro}, 167
\tkzGetPointCoord(⟨A⟩){⟨name of macro⟩}, 167
\tkzGetPoints{A}{B}, 217

\tkzGetPoints{M,N} , 29
\tkzGetPoints{O'}{M'}, 85
\tkzGetPoints, 29, 58, 74–76, 85
\tkzGetPoints: arguments
    {ref1,ref2}, 29
\tkzGetPoints{⟨ref1⟩}{⟨ref2⟩}, 29
\tkzGetPointsz1z2, 87
\tkzGetPoint{⟨ref⟩}, 29
\tkzGetRandPointOn, 21, 102
\tkzGetSecondPoint{A}, 217
\tkzGetSecondPoint{M} , 30
\tkzGetSecondPoint{Tb}, 82
\tkzGetSecondPoint, 30, 74
\tkzGetSecondPoint: arguments
    ref2, 30
\tkzGetSecondPoint{⟨ref2⟩}, 30
\tkzGetVectxy, 56, 57
\tkzGetVectxy: arguments
    (point){name of macro}, 56
\tkzGetVectxy(⟨A, B⟩){⟨text⟩}, 56
\tkzInit, 10, 15, 20, 132
\tkzInit: options
    xmax, 132
    xmin, 132
    xstep, 132
    ymax, 132
    ymin, 132
    ystep, 132
\tkzInit[⟨local options⟩], 132
\tkzInterCC, 29, 94
\tkzInterCC: options
    N, 94
    R, 94
    with nodes, 94
\tkzInterCCN, 94
\tkzInterCCR, 94
\tkzInterCC[⟨options⟩](⟨O, A⟩)(⟨O′, A′⟩) or (⟨O, r⟩)(⟨O′, r′⟩) or (⟨O, A, B⟩) (⟨O′, C, D⟩), 94
\tkzInterLC, 89
\tkzInterLC: options
    N, 89
    R, 89
    with nodes, 89
\tkzInterLC[⟨options⟩](⟨A, B⟩)(⟨O, C⟩) or (⟨O, r⟩) or (⟨O, C, D⟩), 89
\tkzInterLL, 89
\tkzInterLL(⟨A, B⟩)(⟨C, D⟩), 89
\tkzLabelAngle, 152
\tkzLabelAngle: options
    pos, 152
\tkzLabelAngles, 154
\tkzLabelAngles[⟨local options⟩](⟨A,O,B⟩)(⟨A',O',B'⟩)etc., 154
\tkzLabelAngle[⟨local options⟩](⟨A,O,B⟩), 152
\tkzLabelArc, 155
\tkzLabelArc: arguments
    (pt1,pt2,pt3), 155
    label, 155
\tkzLabelArc: options
    pos, 155
\tkzLabelArc[⟨local options⟩](⟨pt1,pt2,pt3⟩){⟨label⟩}, 155

\tkzLabelCircle, 79, 154
\tkzLabelCircle: options
    R, 154
    radius, 154
\tkzLabelCircle[⟨local options⟩](⟨A,B⟩)(⟨angle⟩){⟨label⟩}, 154
\tkzLabelLine(A,B), 152
\tkzLabelLine, 20, 152
\tkzLabelLine: arguments
    label, 152
\tkzLabelLine: options
    pos, 152
\tkzLabelLine[⟨local options⟩](⟨pt1,pt2⟩){⟨label⟩}, 152
\tkzLabelPoint(A){A_1}, 148
\tkzLabelPoint(A,B,C), 149
\tkzLabelPoint, 148
\tkzLabelPoint: arguments
    point, 148
\tkzLabelPoint: options
    TikZ options, 148
\tkzLabelPoints(A,B,C), 148
\tkzLabelPoints, 148, 149
\tkzLabelPoints: arguments
    list of points, 148, 149
\tkzLabelPoints[⟨local options⟩](⟨A_1,A_2,...⟩), 148, 149
\tkzLabelPoint[⟨local options⟩](⟨point⟩){⟨label⟩}, 148
\tkzLabelSegment(A,B){5}, 150, 155
\tkzLabelSegment, 150
\tkzLabelSegment: arguments
    (pt1,pt2), 150
    label, 150
\tkzLabelSegment: options
    pos, 150
\tkzLabelSegments, 151
\tkzLabelSegments[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩), 151
\tkzLabelSegment[⟨local options⟩](⟨pt1,pt2⟩){⟨label⟩}, 150
\tkzMarkAngle, 143, 144, 207
\tkzMarkAngle: options
    arc, 144
    mark, 144
    mkcolor, 144
    mkpos, 144
    mksize, 144
    size, 144
\tkzMarkAngles, 144
\tkzMarkAngles[⟨local options⟩](⟨A,O,B⟩)(⟨A',O',B'⟩)etc., 144
\tkzMarkAngle[⟨local options⟩](⟨A,O,B⟩), 144
\tkzMarkArc, 143
\tkzMarkArc: options
    color, 143
    mark, 143
    pos, 143
    size, 143
\tkzMarkArc[⟨local options⟩](⟨pt1,pt2,pt3⟩), 143
\tkzMarkRightAngle, 144, 145
\tkzMarkRightAngle: options
    german, 145
    size, 145
\tkzMarkRightAngles, 146

\tkzMarkRightAngles[⟨local options⟩](⟨A,O,B⟩)(⟨A',O',B'⟩)etc., 146
\tkzMarkRightAngle[⟨local options⟩](⟨A,O,B⟩), 145
\tkzMarkSegment, 142
\tkzMarkSegment: options
    color, 142
    mark, 142
    pos, 142
    size, 142
\tkzMarkSegments, 142
\tkzMarkSegments[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩), 142
\tkzMarkSegment[⟨local options⟩](⟨pt1,pt2⟩), 142
\tkzProtractor, 163
\tkzProtractor: options
    lw, 163
    return, 163
    scale, 163
\tkzProtractor[⟨local options⟩](⟨O,A⟩), 163
\tkzpttocm, 166
\tkzpttocm: arguments
    (nombre)name of macro, 166
\tkzpttocm(⟨nombre⟩){⟨name of macro⟩}, 166
\tkzSaveBB, 20
\tkzSetUpArc, 21, 172
\tkzSetUpArc: options
    color, 172
    line width, 172
    style, 172
\tkzSetUpArc[⟨local options⟩], 172
\tkzSetUpColors, 169
\tkzSetUpCompass, 21, 173, 174
\tkzSetUpCompass: options
    color, 173
    delta, 173
    line width, 173
    style, 173
\tkzSetUpCompass[⟨local options⟩], 173
\tkzSetUpLabel, 21, 174
\tkzSetUpLine, 21, 108, 171
\tkzSetUpLine: options
    add, 171
    color, 171
    line width, 171
    style, 171
\tkzSetUpLine[⟨local options⟩], 171
\tkzSetUpPoint, 21, 169–171
\tkzSetUpPoint: options
    color, 169
    fill, 169
    shape, 169
    size, 169
\tkzSetUpPoint[⟨local options⟩], 169
\tkzSetUpStyle, 21, 174
\tkzSetUpStyle[⟨local options⟩], 174
\tkzShowBB, 133, 134
\tkzShowBB[⟨local options⟩], 133
\tkzShowLine, 160, 161, 174
\tkzShowLine: options
    K, 160

bisector, 160
gap, 160
length, 160
mediator, 160
orthogonal, 160
perpendicular, 160
ratio, 160
size, 160
\tkzShowLine[⟨local options⟩](⟨pt1,pt2⟩) or (⟨pt1,pt2,pt3⟩), 160
\tkzShowTransformation, 161, 162
\tkzShowTransformation: options
K, 161
gap, 161
length, 161
projection=onto pt1--pt2, 161
ratio, 161
reflection= over pt1--pt2, 161
size, 161
symmetry=center pt, 161
translation=from pt1 to pt2, 161
\tkzShowTransformation[⟨local options⟩](⟨pt1,pt2⟩) or (⟨pt1,pt2,pt3⟩), 161
\tkzTangent, 60

\useasboundingbox, 132
\usetkzobj{all}, 20
\usetkztool, 21

\Vx, 56
\Vy, 56

\xstep, 132

\ystep, 132