

Euclidean Geometry with tkz-euclide

`\tkzInterLC` and `\tkzInterCC`

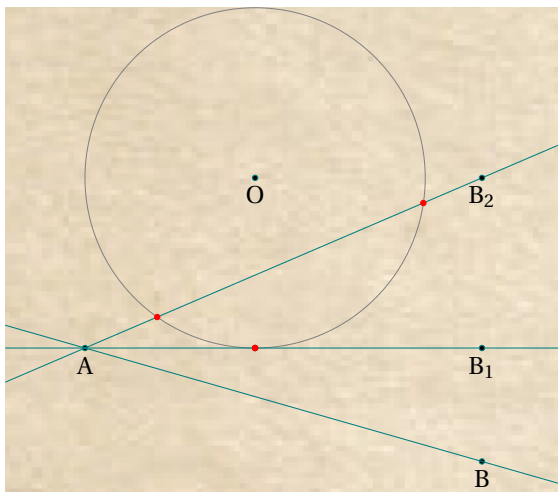
Alain Matthes

July 15, 2022

This document brings together some notes about Intersection of line with a circle and intersection about two circles with tkz-euclide,

`\tkzInterLC`

0.1 Position of a line with respect to a circle



```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x    y    name
                3    /3    /O,
                3    /0    /M,
                0    /0    /A,
                7    /-2   /B}
\tkzDrawCircle(O,M)
\tkzDrawPoints(A,B,O,M)
\tkzLabelPoints(A,B,O)
\tkzDrawLine(A,B)
\foreach \i [count=\j] in {0,3}{%
  \coordinate (B) at (7,\i);
  \tkzLabelPoint(B){$B\_j$}
  \tkzDrawLine(A,B)
  \tkzTestInterLC(A,B)(O,M)
  \iftkzFlagLC
    \tkzInterLC(A,B)(O,M)
    \tkzGetPoints{a}{b}
    \tkzDrawPoints[red](a,b)
  \fi
  \tkzDrawPoints(B)
}
\end{tikzpicture}
```

Figure 1: Position of a line with respect to a circle

Let be a line (AB) and a circle of center O passing through M. The first part will discuss about the position of a line with respect to a circle.

There are three cases: figure 1

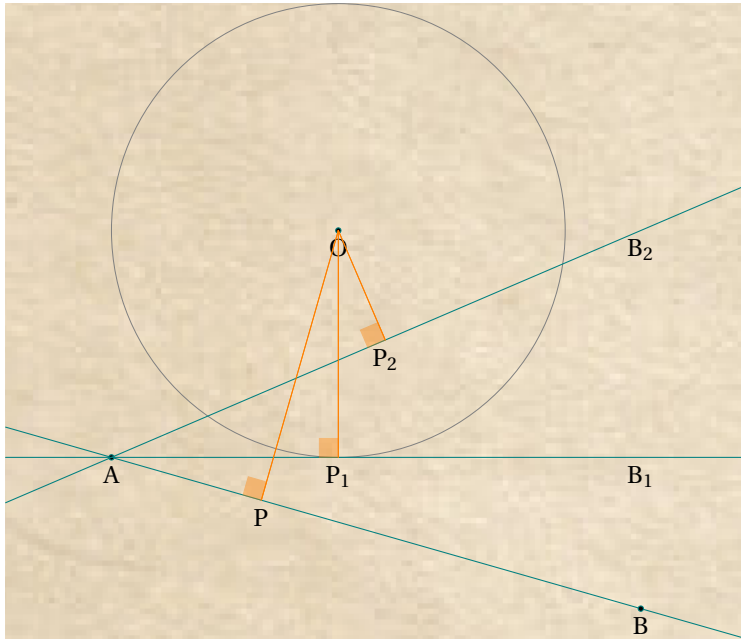
- the line will lie completely outside the circle, like with (A,B);

- the line will touch the circle in exactly one point , like with (A, B_1) ;
- the line must intersect the circle at two distinct points, like with (A, B_2) .

In geometry, a line meeting a circle in exactly one point is known as a tangent line, while a line meeting a circle in exactly two points is known as a secant line.

0.2 How to determine the position of a line with respect to a circle

To determine the position of a line with respect to a circle, all we need to do is find its distance from the center of the circle, and compare it with its radius.



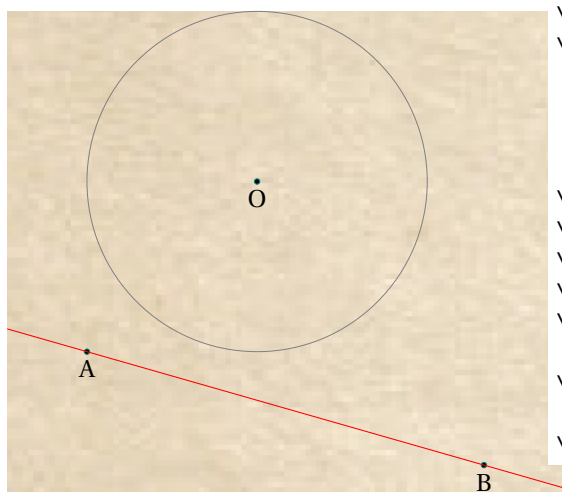
The second one will derive a small expression to obtain the length of a chord intercepted by a circle on a line. The last part will discuss the conditions for a line to become a tangent to a circle.

0.3 How to test?

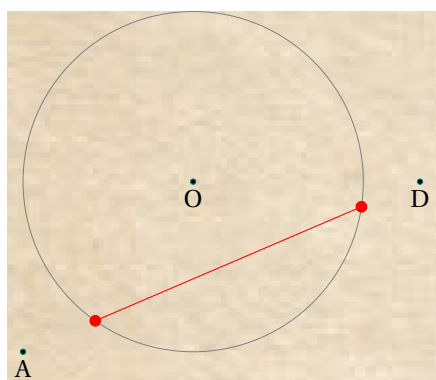
The macro is `\tkzTestInterLC(A,B)(O,M)`. Let's see how this macro is made. We need to create this: `\newif\iftkzFlagLC\tkzFlagLCfalse`

```
\def\tkzTestInterLC(#1,#2)(#3,#4){%
\tkz@Projection(#1,#2)(#3){tkz@pth}% distance center to line
\tkz@@CalcLength(#3,tkz@pth){tkz@mathLen}%
\tkz@@CalcLength(#3,#4){tkzLengthResult}% calculation of the radius
\ifdim\tkz@mathLen pt>\tkzLengthResult pt\relax% comparaison
% Modification of the flag
\tkzFlagLCfalse
\else
\tkzFlagLCtrue
\fi
}
```

- Test with the line (A, B)
- Test with the line (A, D)



```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x    y    name
                3    /3    /O,
                3    /0    /M,
                0    /0    /A,
                7    /-2   /B}
\tkzDrawCircle(O,M)
\tkzDrawPoints(A,B,O)
\tkzLabelPoints(A,B,O)
\tkzTestInterLC(A,B)(O,M)
\iftkzFlagLC
  \else
\tkzDrawLine[red](A,B)
  \fi
\end{tikzpicture}
```



```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x    y    name
                3    /3    /O,
                3    /0    /M,
                0    /0    /A,
                7    /3    /D}
\tkzDrawCircle(O,M)
\tkzDrawPoints(A,D,O)
\tkzLabelPoints(A,D,O)
\tkzTestInterLC(A,D)(O,M)
\iftkzFlagLC
  \tkzInterLC(A,D)(O,M) \tkzGetPoints{I}{J}
\tkzDrawSegment[red](I,J)
\tkzDrawPoints[red,size=4](I,J)
\else
\tkzDrawLine[green](A,D)
\fi
\end{tikzpicture}
```

The macro `\tkzTestInterLC` can be used to check and set a flag `tkzFlagLC`, then the flag was available and is being set to "false" if the intersection is empty and in other cases `tkzFlagLC` is set to "true".

The macro `\iftkzFlagLC` allows you to act according to the result.

0.4 Choice of points

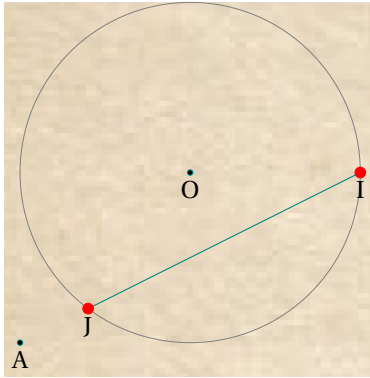
0.4.1 One of the points is known

In the following example the known and common point is the point M. The second point of intersection which will be named N will be obtained with the macro `\tkzGetPoint`

0.4.2 The line passes through the center of the circle

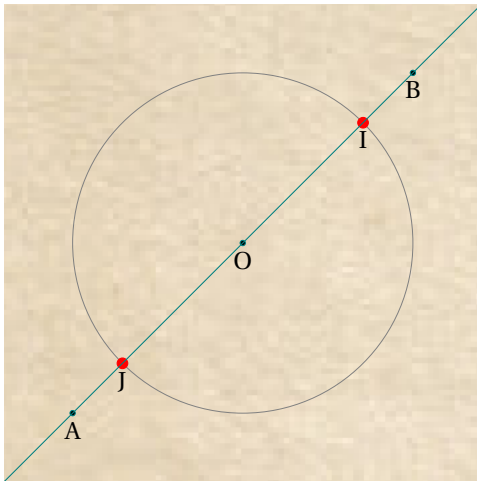
If the line passes through the center of the circle and moreover through a known and common point then it is better to avoid looking for the intersection but for example to use a symmetry.

You can then use the "near" option. The point obtained with `\tkzGetFirstPoint` is the closest point to the first point that defines the line. It cannot be the center which is equidistant from



```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x    y    name
                3    /3    /O,
                6    /3    /I,
                0    /0    /A}
\tkzDrawCircle(O,I)
\tkzDrawPoints(A,O)
\tkzLabelPoints(A,O)
\tkzInterLC[common=M](A,I)(O,J)
\tkzGetFirstPoint{J}
\tkzDrawSegment(I,J)
\tkzDrawPoints[red,size=4](I,J)
\tkzLabelPoints(I,J)
\end{tikzpicture}
```

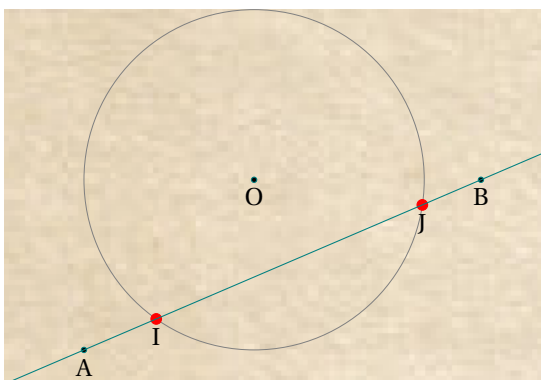
any point of the circle. Bien évidemment vous devez écrire `\tkzInterLC[near](B,0)(0,M)` et non `\tkzInterLC[near](0,B)(0,M)`



```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x    y    name
                3    /3    /O,
                3    /0    /M,
                0    /0    /A,
                6    /6    /B}
\tkzDrawCircle(O,M)
\tkzDrawPoints(A,B,O)
\tkzLabelPoints(A,B,O)
\tkzInterLC[near](B,A)(O,M)
\tkzGetPoints{I}{J}
\tkzDrawPoints[red,size=4](I,J)
\tkzDrawLine(A,B)
\tkzLabelPoints(I,J)
\end{tikzpicture}
```

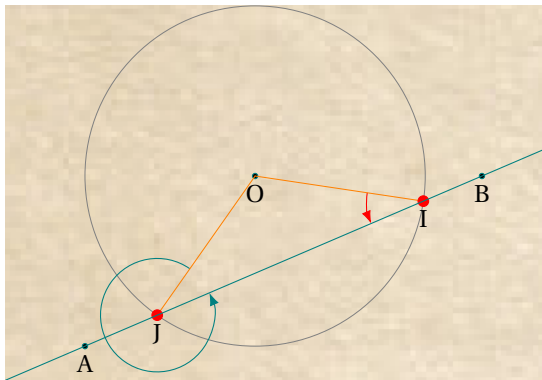
0.4.3 General case

First possibility If you know a point near an intersection then you can use the option **near**.



```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x    y    name
                3    /3    /O,
                3    /0    /M,
                0    /0    /A,
                7    /3    /B}
\tkzDrawCircle(O,M)
\tkzDrawPoints(A,B,O)
\tkzLabelPoints(A,B,O)
\tkzInterLC[near](A,B)(O,M)
\tkzGetPoints{I}{J}
\tkzDrawPoints[red,size=4](I,J)
\tkzDrawLine(A,B)
\tkzLabelPoints(I,J)
\end{tikzpicture}
```

The second possibility, only if the line does not pass through the center. The algorithm that handles the intersection compares the following two angles: $\widehat{I,J,O}$ and $\widehat{J,I,O}$. `TkzFirstPointResult` is assigned to the smallest angle. Angles with `tkz-euclide` are measured with the macro `\tkzFindAngle` always positively between 0° and 360° .



```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{0/0/O,5/1/A,2/2/B,3/1/D}
\tkzDefPoints{% x      y      name
               3      /3      /O,
               3      /0      /M,
               0      /0      /A,
               7      /3      /B}
\tkzDrawCircle(O,M)
\tkzDrawPoints(A,B,O)
\tkzLabelPoints(A,B,O)
\tkzInterLC(A,B)(O,M) \tkzGetPoints{J}{I}
\tkzMarkAngle[red,-Latex](O,I,I)
\tkzMarkAngle[teal,-Latex](O,J,I)
\tkzDrawSegments[new](I,O O,J)
\tkzDrawPoints[red,size=4](I,J)
\tkzDrawLine(A,B)
\tkzLabelPoints(I,J)
\end{tikzpicture}
```

As you can see, $\widehat{I}$ is **clockwise** while $\widehat{J}$ is **counterclockwise**.

Remark: How to test if a point is on a line.

`\tkzInterCC`

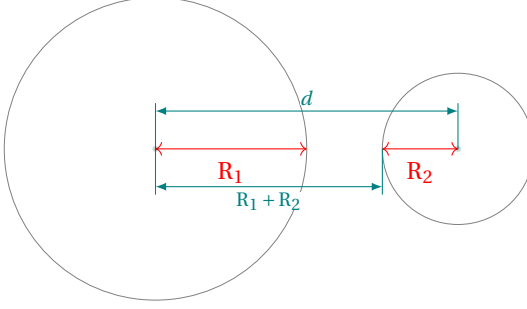
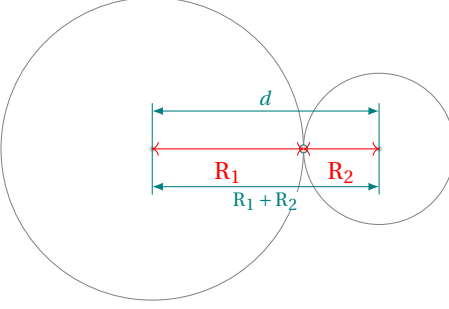
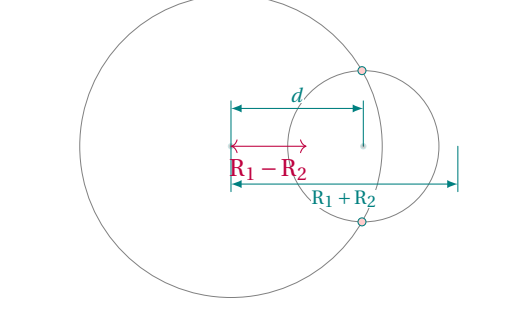
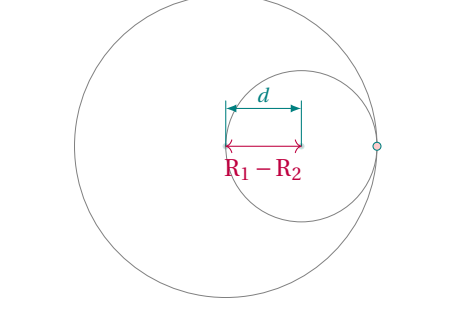
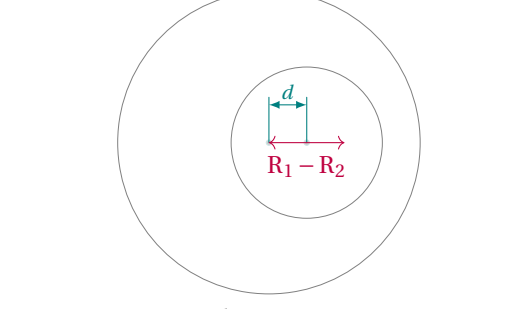
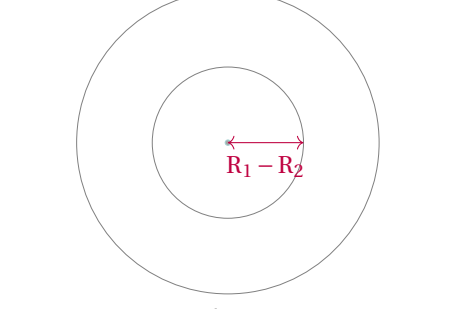
0.1 Relative position of two circles

- Common center. Concentric circles the distance between centers equal to zero. They have no common points. One circle lying inside another;
- Eccentric interior the distance between their centers is greater than 0 and less than the difference of their radius. One circle lying inside another;
- Internally tangent Circles touch inside. they have a common point. The distance between their centers is equal to the difference of their radius. Touching internally
- Externally tangent, They have just one common point and all other points are exterior to the other. The distance between their centers equals the sum of their radius; Touching externally
- Noncrossing circles or Externals have no common points. they do not overlap. If the distance between the centers of the circles is greater than zero and greater than the sum of the radii, then one circle lies outside another one; Externals, if they do not overlap and the distance between their centers is greater than the sum of their radius. Two circles lying outside each other
- Crossing circles have two common points If the distance between the centers of circles is greater than zero, and its value is between the difference and the sum of radii, then the circles cross in two points. they intersect at two different points and the distance between their centers is less than the sum of their radius. No matter if they have the same or different radius. Intersecting at two points

0.2 How to determine relative position of two circles

Let be two circles $\mathcal{C}_\infty$ and $\mathcal{C}_\epsilon$ having as center O_1 and radius R_1 for the first one, O_2 and radius R_2 for the second one. Let d be the distance between O_1 and O_2 and suppose that $R_1 > R_2$.

- $d > R_1 + R_2$ Noncrossing circles or Externals
- $d = R_1 + R_2$ Externally tangent item $R_1 - R_2 < d < R_1 + R_2$ Crossing circles
- $d = R_1 - R_2$ Internally tangent
- $d < R_1 - R_2$ Eccentric circles
- $d = 0$ Concentric circles

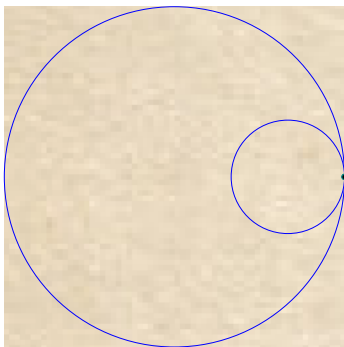
 $d > R_1 + R_2$ Lying outside each other Noncrossing circles or Externals	 $d = R_1 + R_2$ Touching externally Externally tangent
 $R_1 - R_2 < d < R_1 + R_2$ Intersecting at two points Crossing circles	 $d = R_1 - R_2$ Touching internally Internally tangent
 $d < R_1 - R_2$ One lying inside other Eccentric circles	 $d = 0$ One lying inside another Concentric circles

0.3 How to test?

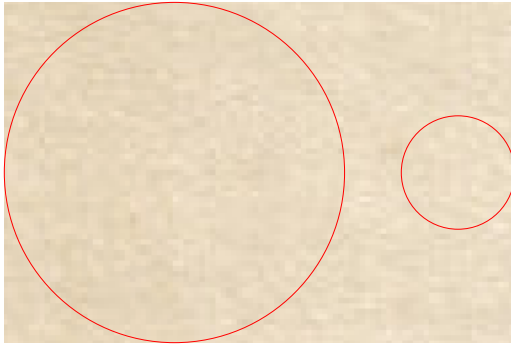
We have to determine if $R_1 - R_2 \leq d \leq R_1 + R_2$ which is the condition for the intersection to be non empty.

```
% test avec des nodes  R-r <= d <= R+r
\def\tkzTestInterCC(#1,#2)(#3,#4){%
\tkz@@CalcLength(#1,#3){tkz@mathLen}% distance entre les centres
\tkz@@CalcLength(#2,#1){tkz@rA}%calcul du rayon
\tkz@@CalcLength(#4,#3){tkz@rB}%calcul du rayon
% test if d <= rA + rB ?
\edef\tkz@rS{\fpeval{\tkz@rA+\tkz@rB}}
\ifdim\tkz@mathLen pt > \tkz@rS pt\relax%
\tkzFlagCCfalse
\else
% now test if d>= rA - rB or rB-rA
\ifdim #2 pt > #1 pt\relax%
\edef\tkz@rD{\fpeval{\tkz@rA-\tkz@rB}}
\else
\edef\tkz@rD{\fpeval{\tkz@rB-\tkz@rA}}
\fi
\ifdim \tkz@rD pt > \tkz@mathLen pt\relax%
\tkzFlagCCfalse \else \tkzFlagCCtrue \fi
}
```

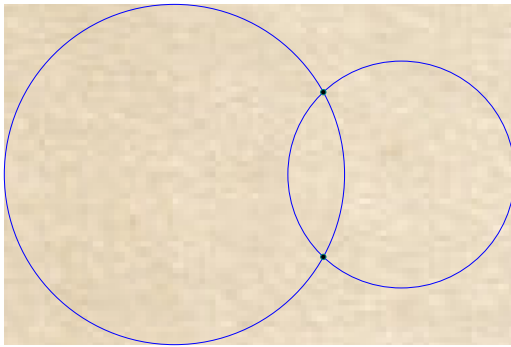
0.3.1 Test : relative position of two circles



```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x    y    name
0    /0    /0,
3    /0    /A,
2    /0    /O'}
\tkzTestInterCC(0,A)(O',A)
\iftkzFlagCC
\tkzDrawCircles[blue](0,A O',A)
\tkzInterCC(0,A)(O',A)
\tkzGetPoints{I}{J}
\tkzDrawPoints(I,J)
\else
\tkzDrawCircles[red](0,A O',A)
\fi
\end{tikzpicture}
```

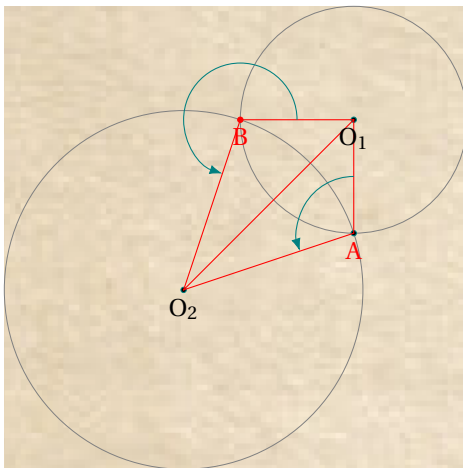


```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x      y      name
                0      /0      /O,
                3      /0      /A,
                4      /0      /B,
                5      /0      /O'}
\tkzTestInterCC(O,A)(O',B)
\iftkzFlagCC
\tkzDrawCircles[blue](O,A O',B)
\else
\tkzDrawCircles[red](O,A O',B)
\fi
\end{tikzpicture}
```



```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x      y      name
                0      /0      /O,
                3      /0      /A,
                2      /0      /B,
                4      /0      /O'}
\tkzTestInterCC(O,A)(O',B)
\iftkzFlagCC
\tkzDrawCircles[blue](O,A O',B)
\tkzInterCC(O,A)(O',B)
\tkzGetPoints{I}{J}
\tkzDrawPoints(I,J)
\else
\tkzDrawCircles[red](O,A O',B)
\fi
\end{tikzpicture}
```

0.3.2 Crossing circles common point

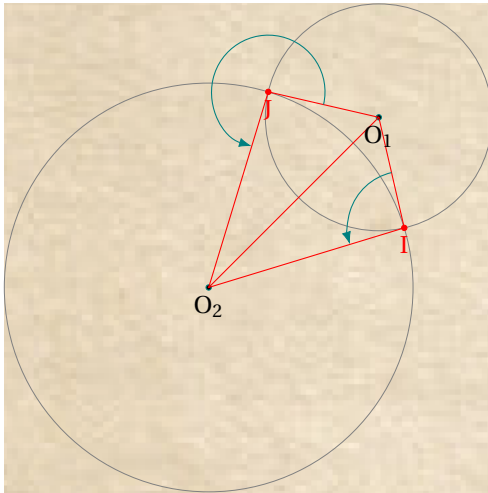


```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x      y      name
                3      /3      /O_1,
                3      /1      /A,
                0      /0      /O_2}
\tkzDrawCircles(O_1,A O_2,A)
\tkzInterCC[common=A](O_1,A)(O_2,A)
\tkzGetFirstPoint{B}
\tkzDrawPoints(O_1,O_2)
\tkzDrawPoints[red](B)\tkzDrawPoints(A)
\tkzDrawSegments[red](O_1,O_2 O_1,A O_1,B
O_2,A O_2,B)
\tkzMarkAngle[teal,-Latex](O_1,A,O_2)
\tkzMarkAngle[teal,-Latex](O_1,B,O_2)
\tkzLabelPoints[red](A,B)
\tkzLabelPoints(O_1,O_2)
\end{tikzpicture}
```

0.3.3 Crossing circles general case

The algorithm that handles the intersection compares the following two angles: $\widehat{O_1, I, O_2}$ and $\widehat{O_1, J, O_2}$. **TkzFirstPointResult** is assigned to the smallest angle. Angles with **tkz-euclide** are measured

with the macro `\tkzFindAngle` always positively between 0° and 360° . As you can see, $\hat{I}$ is **clockwise** while $\hat{J}$ is **counterclockwise**.



```
\begin{tikzpicture}[scale=.75,old paper]
\tkzDefPoints{% x    y    name
                3    /3    /O_1,
                3    /1    /A,
                0    /0    /O_2,
                2    /3    /B}
\tkzDrawCircles(O_1,A O_2,B)
\tkzInterCC(O_1,A)(O_2,B) \tkzGetPoints{I}{J}
\tkzDrawPoints(O_1,O_2)
\tkzDrawPoints[red](I,J)
\tkzDrawSegments[red](O_1,O_2 O_1,I O_1,J O_2,I O_2,J)
\tkzMarkAngle[teal,-Latex](O_1,I,O_2)
\tkzMarkAngle[teal,-Latex](O_1,J,O_2)
\tkzLabelPoints[red](I,J)
\tkzLabelPoints(O_1,O_2)
\end{tikzpicture}
```

Algorithm used for `\tkzInterLC`

Calculate the intersection of a ray and a sphere

The line segment is defined from p_1 to p_2

The sphere is of radius r and centered at sc

There are potentially two points of intersection given by

$$p = p_1 + \mu_1 (p_2 - p_1)$$

$$p = p_1 + \mu_2 (p_2 - p_1)$$

Return FALSE if the ray doesn't intersect the sphere.

*/

```
int RaySphere(XYZ p1,XYZ p2,XYZ sc,double r,double *mu1,double *mu2)
```

```
{
```

```
double a,b,c;
```

```
double bb4ac;
```

```
XYZ dp;
```

```
dp.x = p2.x - p1.x;
```

```
dp.y = p2.y - p1.y;
```

```
dp.z = p2.z - p1.z;
```

```
a = dp.x * dp.x + dp.y * dp.y + dp.z * dp.z;
```

```
b = 2 * (dp.x * (p1.x - sc.x) + dp.y * (p1.y - sc.y) + dp.z * (p1.z - sc.z));
```

```
c = sc.x * sc.x + sc.y * sc.y + sc.z * sc.z;
```

```
c += p1.x * p1.x + p1.y * p1.y + p1.z * p1.z;
```

```
c -= 2 * (sc.x * p1.x + sc.y * p1.y + sc.z * p1.z);
```

```
c -= r * r;
```

```
bb4ac = b * b - 4 * a * c;
```

```
if (ABS(a) < EPS || bb4ac < 0) {
```

```
    *mu1 = 0;
```

```
    *mu2 = 0;
```

```
    return(FALSE);
```

```
}
```

```
*mu1 = (-b + sqrt(bb4ac)) / (2 * a);
```

```

    *mu2 = (-b - sqrt(bb4ac)) / (2 * a);

    return(TRUE);
}

```

Algorithm used for\tkzInterCC

circle_circle_intersection()
Determine the points where 2 circles in a common plane intersect.

```

int circle_circle_intersection(
    // center and radius of 1st circle
    double x0, double y0, double r0,
    // center and radius of 2nd circle
    double x1, double y1, double r1,
    // 1st intersection point
    // 2nd intersection point

```

This is a public domain work. 3/26/2005 Tim Voght

```

int circle_circle_intersection(double x0, double y0, double r0,
    double x1, double y1, double r1,
    double *xi, double *yi,
    double *xi_prime, double *yi_prime)
{
    double a, dx, dy, d, h, rx, ry;
    double x2, y2;
    dx and dy are the vertical and horizontal distances between
    the circle centers.
    dx = x1 - x0;
    dy = y1 - y0;
    Determine the straight-line distance between the centers.
    d = sqrt((dy*dy) + (dx*dx));
    d = hypot(dx,dy); // Suggested by Keith Briggs
    Check for solvability.
    if (d > (r0 + r1))
    {
        no solution. circles do not intersect.
        return 0;
    }
    if (d < fabs(r0 - r1))
    {
        no solution. one circle is contained in the other
        return 0;
    }
    'point 2' is the point where the line through the circle
    intersection points crosses the line between the circle
    centers.
    Determine the distance from point 0 to point 2.
    a = ((r0*r0) - (r1*r1) + (d*d)) / (2.0 * d) ;

```

```

Determine the coordinates of point 2.
x2 = x0 + (dx * a/d);
y2 = y0 + (dy * a/d);

Determine the distance from point 2 to either of the
intersection points.
h = sqrt((r0*r0) - (a*a));
Now determine the offsets of the intersection points from
point 2.
rx = -dy * (h/d);
ry = dx * (h/d);
Determine the absolute intersection points.
xi = x2 + rx;
xi_prime = x2 - rx;
yi = y2 + ry;
yi_prime = y2 - ry;
return 1;
}

```