Circle Inversions
fun with a serious undertone

Abstract
Circle inversions are exercised and drawn with PostScript operators which are also included in this plain TEX document. Interesting pictures will be shown, resulting from inversion of straight line pieces and other procedures. I demonstrate a way to calculate the circle of anti-similitude, by which two circles are inverses of each other. Furthermore, I show how one can transform two distinct circles into two concentric circles. How to draw a circle orthogonal to a circle which passes through one or two points within the circle is done via the circle inversion technique. The above is generalized into finding the circle which cuts the boundary at an arbitrary angle, e.g. 80 degrees, and passes through a point within the circle. Orthogonal circular arcs can form an Escher-like grid, as he used in his Circle Limits. Four variants of the grid of Circle Limits III have been included. The first cuts the boundary at 80 degrees, the second at 90 degrees, and the third with a mixture of both. The fourth is Coxeter's solution. A smiley pattern is inverted in (orthogonal) circular arcs within a circle with the aid of PostScript's `pathforall` by (repeated use of) circle inversion. How to draw a circle orthogonal to 1, 2 or 3 other distinct circles is shown. Apollonius problem is solved by the use of the circle inversion transformation and also by transforming the 3 quadratic equations into 1 non-linear equation and a 2x2 system of linear equations, and solving these equations in PostScript and MetaPost. A closer look yielded that we only have to solve one quadratic equation in r, the radius of the wanted circle, in order to obtain the solution of Apollonius problem. Coding problems in MetaPost will be mentioned and circumvented. I demonstrate the way one can create and use a PostScript library. A plea is made for creating and maintaining a PostScript library of operators, graphics and utilities. A snapshot of this growing library is included. A few tiny but handy PostScript operators are given next to a (numerical) PostScript operator to solve a 3x3 linear system of equations, where partial pivoting is implemented and the calculations are done with the accuracy of the underlying computer arithmetic, which is much better than MetaPost's accuracy for the moment. How to overload a PostScript operator, e.g. `length`, is given. The question whether the PostScript library can be used in MetaPost will be answered. The pearl of the paper is twofold: first the rediscovery that Apollonius problem is solved by the solution of a quadratic equation, and second the operator `Apollonius`, which reflects this rediscovery and can be used to obtain all 8 solutions of Apollonius problem. Another gem is `Apollonius2`, which is suited for the case that one circle contains the other two. The culmination of it all is the operator `radical` for drawing the radical circle of three given distinct circles.

Keywords
Apollonius, Cabri, circle inversion, circle covered by touching circles, circle limit, circle of anti-similitude, Coxeter, Descartes circle theorem, Escher, Java, Mathematica, Metafont, MetaPost, minimal markup, mixed-language programming, orthogonal circles, overloading polymorphic operator, Peaucellier-Lipkin linkage, (partial) pivoting, plain TEX, PostScript library, radical circle, reflection, Rerich, Sandaku, Soddy, solving 3X3 linear equations

Introduction

While familiarizing myself with hyperbolic geometry I needed to draw a circle orthogonal to a given circle such that the orthogonal circle also passes through one or two prescribed points within or on the given circle. This led me to the circle inversion technique, which was invented by L.I. Magnus in 1841. Circle inversion is considered as an inroad to higher mathematics. In this note plane and analytic geometry at the level I learned at high school is used.

Circle inversions are about inverting points, lines, circles and, in general, patterns in a circle.

In the following the various inversions are programmed as PostScript operators, because PostScript has the `arc` operator for circular arcs, which is much more powerful than Hobby's `fullcircle` definition in MetaPost, and because PostScript is a widespread, time-proven graphical language, which abstracts from the printing device. Another useful feature of PostScript is that one can transform the user space independently from the device space. Nice is also that one can transform fonts. Moreover, PostScript enjoys the accuracy of the underlying computer arithmetic, which is better than the accuracy obtained by MetaPost for the moment. PostScript is handy in a workflow and the resulting graphics can be included in AnyTEX or troff documents.[1] For interactive, animated graphics Java might be the tool to use, but I have no hands-on experience with this as yet.

For inclusion of the PostScript graphics in `pdf(La)`TEX, the pictures must be converted to a `pdf`TEX friendly format, for example to (trimmed) `.pdf`, alas.[2]

A few operators use the stack only. Most of the operators use 'local' variables created in dictionaries associated with the names of the operators. As a consequence there are no name conflicts and therefore the operators can be collected in a library for reuse.

Don Lancaster provides his so-called Gonzo PostScript utilities on the internet. Don likes PostScript even more than I do, so it seems. The operators in Adobe's PostScript Tutorial and Cookbook, the blue book, have been used as a starting point for my PostScript library. The PostScript operators in my notes from more than a decade ago: *Tiling in PostScript and MetaFont, MAPS97.2*, and *Paradigms: Just a little bit of PostScript, MAPS96.2* (rev 1997), already formed a library in status nascendi, as I mentioned when I launced my BLUe collection, but ... at the time I was not aware of the way to fruitfully use PostScript's `run` operator for inclusion of the library.

Undoubtedly there is a wealth of PostScript operators out there scattered over the WWW. Why not collect, test and put them together into a library for reuse?

To those who shrug their shoulders and pass by *programming* in PostScript with the argument that it is too low-level:

"YES, but ..."

we have abstraction as our powerful mental tool, and we can build higher-level operators on top of the basic ones.[3] Moreover, we can use PostScript in MetaPost, to a certain extent.

For example: what is wrong with determining the intersection point of two lines or (sets of geometric loci) by an operator `intersect`, which assumes on the stack the data which characterizes the two lines, i.e. two points for each line, and leaves on the stack the point of intersection? With respect to the parameter passing in other languages the variation by communicating via the stack and storing the values in local variables, I consider this syntactic sugar, not to be confused with the use of the stack only.

This 3-in-1 paper touches on math, graphics, computation, numerical analysis, programming, and it consists of the parts: Circle Inversions, Apollonius problem, Orthogonal Circles, and the use and creation of a PostScript library.

Notation

Capital letters are used to denote points, lines and circles, with the circles in italics. A capital superscripted by i denotes the inversion. Subscript r denotes the radius of the circle, parenthesized subscripts (x, y) denote the coordinates of the circle centre: $I_{r,(x,y)}$ denotes the inversion circle with radius r and centre (x, y); $\mathrm{I}_{r,(x,y)}$ denotes the centre of the circle I. In the operator code what should be supplied on the stack is documented after %-signs and separated by `==>` from what is delivered on the stack, in the PostScript documentation tradition. An angle is denoted by $\angle$. A triangle by $\triangle$. For similarity the symbol $\sim$ is used.

The inversion circle is dashed [1 2] ([n m] means à la PostScript: n on (black) m off (white), cyclically) with linethickness .25 (`invcir`), the inversions are dashed [1 1] with linethickness 1 (`invlin`), auxiliary lines are dashed [1 3] with linethickness .25 (`auxlin`), the wanted circles are bold, i.e. have linethickness 1.5.[4] Inversion circles as auxiliaries are dashed [1 3] with linethickness .75 (`Auxlin`). The dashing conventions are imposed by printing in black and white.

Requirements for a PostScript library

It would be nice if a PostScript library would consist of well-documented, mean-and-lean, legible, and robust[5] PostScript operators. Next to operators the library might contain PostScript pictures, utilities (as mentioned by M. Gelderman MAPS19, 1997, to be found on the CTAN in the directory /support/psutils) ...

The TEX-world uses MetaPost, so one can ask oneself: can the PostScript library be used in MetaPost? Yes ... see Appendix II.

Use of the PostScript library on a Windows system can be done by inclusion in your PostScript code

```
(C:\\PSlib\\PSlib.eps) run
```

provided that the file `PSlib.eps` is stored on the C disk in the directory `PSlib`. The documentation of the `run` operator in Adobes' reference manual, the red book, is not explicit on this feature.[6]

The above is analogous to TEX's `\input` and MetaFont's and MetaPost's `input` for (library) macro definitions.

Outdated Math books

While working on this note, I realized that Math books which contain construction methods based on ruler and compass are outdated, because everybody owns a PC nowadays with their graphical user interfaces and powerful (graphical) software.

Cabri software

Nice is the (commercial) Cabri software which provides you with an interactive experimenting environment. Interesting is Wilson's Inverse Geometry WWW, which uses Cabri Java: http://www.maths.gla.ac.uk/~wws/cabripages/inversive/inversive0.html
The difference with this work is that below operators are provided in batch oriented PostScript, while the Java applets from e.g. Cabri, as used by Wilson, are interactive and facilitate animation.

Mathematica

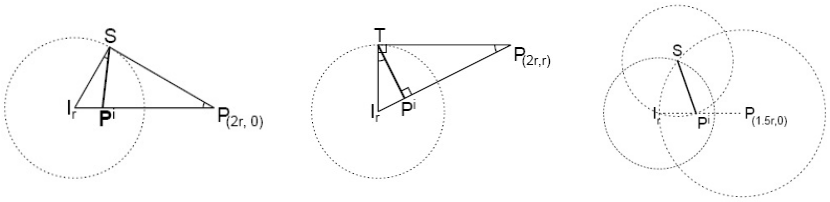
Professional software for graphics (and animation as well) is the commercial Mathematica. A free Mathematica reader can be downloaded, however, which ensures that one can view and animate work of others, the so-called mathematica notebooks, to start with Mathematica's own demos: a mer à boire.
See http://mathworld.wolfram.com/Inversion.html for a Mathematica Inversion notebook.

Point inversion

Inversion of a point P in a circle $I_{r,(x,y)}$ is that point P^i on the line IP defined by $|IP^i|\cdot|IP| = r^2$. Explicitly

$$\vec{P}^i = \vec{I}_r + \frac{\vec{P} - \vec{I}_r}{|\vec{P} - \vec{I}_r|^2} r^2.$$

Geometrical ruler and compass constructions are shown below. However, for practical purposes they are no longer necessary, because of the PostScript operators, based on plane and analytic geometry.



Proof of the construction left: $\triangle I_r P^i S \sim \triangle I_r SP$.
Proof of the construction in the middle: $r^2 = IT^2 = IP^i \cdot IP$.
Proof of the construction right:[7] since $\triangle IPS \sim \triangle ISP^i$, $IP : r = r : IP^i$.

Remark In the right picture we need to determine the intersection point S of the 2 circles. We could solve 2 quadratic equations in 2 unknowns for this; simpler is to exercise plane geometry[8] within the isosceles $\triangle SPI$, with base r and sides d, the distance between I and P.

Properties

- ☐ Points on the inversion circle are invariant.
- ☐ Points outside the inversion circle are mapped onto points inside the circle and vice versa.
- ☐ The point at infinity is the inversion of the centre of the inversion circle.
- ☐ The inversion point is geometrically given by the intersection of the line IP with the chord which connects the tangent points T.[9] (See above middle picture.)
- ☐ Distances are not preserved.
- ☐ The mapping is anti-conformal, i.e. angles are preserved but the orientation is reversed.

PS code The order in which to put the parameters on the stack must be chosen. I chose to put first the object which has to be inverted and second the inversion circle. This can be handy for the case that the inverted object has to be inverted again, for example inversion and back, or repeatedly as in the section inversion of a smiley pattern for the 2^{nd} level inversions.

```
/pointinversion
% Px, Py:    point to be inverted
% Ix, Iy, r: centre and radius of the inversion circle
%==>
% px, py:    inverted point
{0 begin                                %open the local dictionary
/r exch def  /Iy exch def  /Ix exch def %collect values from the stack
/Py exch def /Px exch def               %LIFO in the (local) dictionary
/Px Px Ix sub def /Py Py Iy sub def     %shift origin to centre of circle I
/factor r Px Py size div dup mul def
/px Ix factor Px mul add def /py Iy factor Py mul add def
px py                                   %put solution on the stack
end                                     %close pointinversion dictionary
} def
/pointinversion load 0 10 dict put
```

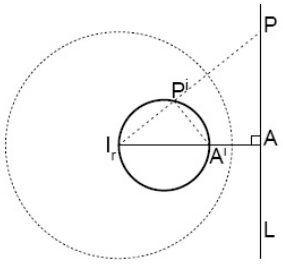
Line inversion

Properties

- ☐ The inversion of a line through the inversion centre is the line itself.
- ☐ The inversion of a line, not through the centre of the inversion circle, is a circle through the centre of the inversion circle, and vica versa.

Proof (Courtesy Courant&Robbins: *What is mathematics?*)

Drop a perpendicular from the centre I_r to the straight line L, see the picture below.

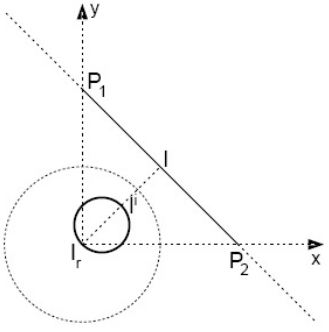


Let A be the point where the perpendicular meets L, and let A^i be the inverse point of A. Mark any point P on L, and let P^i be its inverse point. Since $|IA^i| \cdot |IA| = |IP^i| \cdot |IP| = r^2$, it follows that

$$\frac{|IA^i|}{|IP^i|} = \frac{|IP|}{|IA|}$$

Hence $\triangle IP^iA^i \sim \triangle IAP$, $\angle IP^iA^i$ is a right angle. P^i lies on the circle with I_rA^i as diameter.

Back to the line inversion problem.



The picture is rotation invariant. So the picture can be rotated around I_r, the centre, such that the line P_1P_2 will be perpendicular to the x-axis. The angle of rotation for P_1 in the first quadrant is

$$\phi = \arctan \frac{P_{1y} - P_{2y}}{P_{1x} - P_{2x}}$$

which is adjusted in the implementation successively by[10]

$$\phi := \phi - 180 \qquad \text{if} \quad 180^\circ < \phi < 360^\circ$$
$$\phi := 90 - \phi$$

such that P_1 in all 4 quadrants is accounted for.

Calculation of the inverted circle

Drop the perpendicular from the centre of the inversion circle on the line through P_1 and P_2.[11] Let l be the point where the perpendicular meets P_1P_2 and let us denote the inversion of l by l^i. The centre I_r and the point l^i form 2 diametrical points of the inverted circle. The centre of the inverted circle is therefore $.5l^i$, with $|.5l_i|$ the radius of the inverted circle.

The inversion of a line is programmed by the following steps

- shift the centre of the inversion circle to (0,0), meaning shift P_1 and P_2
- rotate the shifted P_1 (and P_2) such that the line P_1P_2 is vertical
- invert the x coordinate of the rotated P_1, and call this point l^i

□ $|.5l^i|$ is the radius of the inverted circle
□ rotate $(.5l^i,0)$ back, in order to obtain the centre of the inverted circle
□ shift the centre of the inverted circle back.

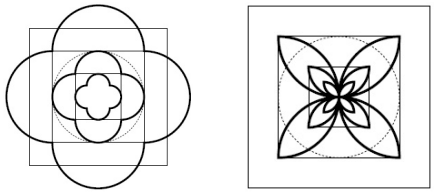
In my earlier MetaFont version I used the equation solving possibilities of MetaFont by specifying equations for the footpoint of the perpendicular from I on the line P_1P_2. The use of rotation is simpler and elegant.

```
/lineinversion
% x1 y1 x2 y2: the points which determine the line
% mx my r:     the centre and radius of the inversion circle
%==>
% mix miy ri:  centre and radius of the inverted line
{0 begin
/eps .0001 def /r exch def
/my exch def /mx exch def
/y2 exch def /x2 exch def
/y1 exch def /x1 exch def
/angle y1 y2 sub x1 x2 sub atan def
angle abs  90 sub abs eps lt
angle abs 270 sub abs eps lt or
{ % special case: vertical line
  x1 my mx my r pointinversion mx my mean
  /myi exch def /mxi exch def
  /ri mxi mx  sub myi my sub size def
}{angle abs 180 sub abs eps lt
  angle abs 360 sub abs eps lt or
  angle abs eps lt or
  { % special case: horizontal line
    mx y1 mx my r pointinversion mx my mean
    /myi exch def /mxi exch def
    /ri mxi mx  sub myi my sub size def
   }{ % general case
    angle 180 gt {/angle angle 180 sub def} if     % reduce to range 0-180
    /angle 90 angle sub def
    /y1 y1 my sub def /x1 x1 mx sub def            % shift point 1
    x1 y1 angle rot pop /lx exch def
    lx abs eps lt{(Line through origin?) print} if % warning
    /mix r dup mul lx div 2 div def /miy 0 def     % center circle
    /ri miy my sub mix mx sub size def             % radius circle
    % rotate back and shift
    mix miy angle neg rot
    /miy exch my add def
    /mix exch mx add def
  }ifelse
}ifelse
mix miy ri
end} def
/lineinversion load 0 25 dict put
```

The special cases, lines parallel to x-axis and y-axis, have been treated separately, no rotation needed. A warning is given when the line passes through the origin, because the line itself is the result and not a circle.

Inversion of line pieces

As example three squares with sides r, 2r , 4r centred around the origin are inverted in the (dashed) circle $I_{r,(0,0)}$. The inversions are drawn in the same picture non-dashed and bold. The right figure is the complement: the inversions of the lines with the sides of the squares left out.



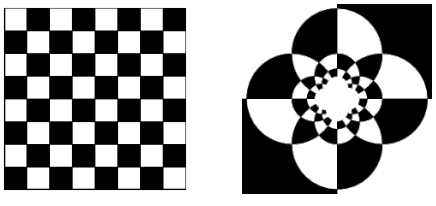
We could extend the above to concentric pentagons, hexagons, but … I don't expect new beautiful results: instead of 4 circular arcs we will obtain 5, 6….

The example is a test for `linepieceinversion`, though it does not exercise the shift of origin.

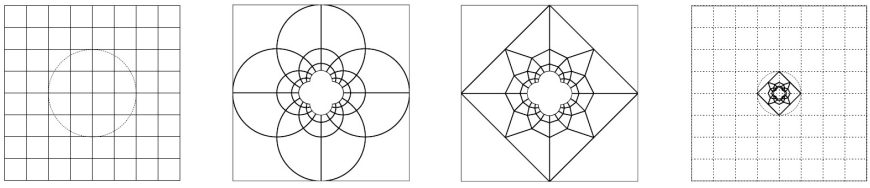
```
/linepieceinversion
% x1 y1 x2 y2: the points which determine the line
% mx my r:     the centre and radius of the inversion circle
%==>
%              path of inverted cirle arc
{0 begin
/r exch def /my exch def /mx exch def
/y2 exch def /x2 exch def /y1 exch def /x1 exch def
x1 y1 x2 y2 mx my r lineinversion  /ri exch def /miy exch def /mix exch def
x1 y1 mx my r pointinversion /y1 exch def /x1 exch def
x2 y2 mx my r pointinversion /y2 exch def /x2 exch def
/phi1 y1 miy sub  x1 mix sub atan def
/phi2 y2 miy sub x2 mix  sub atan def
%(x1, y1)--(0, 0) right from (xm, ym)--(0, 0)?
/psi1 y1 x1 atan def
/psim miy mix atan def
psi1 180 gt {/psi1 psi1 360 sub def} if
                           %correction if xm in 1st and x1 in 4th quadrant
psi1 psim lt
{x1 y1 moveto mix miy ri phi1 phi2 arc}
{x2 y2 moveto mix miy ri phi2 phi1 arc} ifelse
end} def
/linepieceinversion load 0 16 dict put
```

It was troublesome to draw the correct part of the circle without clipping.

A beautiful illustration of the inversion of line pieces is the inversion of a chessboard centred at (0, 0) in a small circle also centred at (0, 0) (M. Gardner 1984, pp. 244-245; R. Dixon 1991). In the picture below the chessboard and its inversion are shown, borrowed from http://mathworld.wolfram.com/Inversion.html.

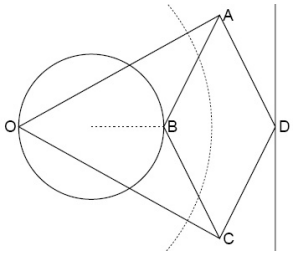


Variation of the inversion of a chessboard is a stainless glass impression of the 'inverse' of a grid of 64 squares, as displayed below. In the inversions the dashed surrounding square indicates ∞, the inverse of the centre.



Explanation The first picture from the left shows the grid and the inversion circle. The second picture shows the inversion of the grid. In the third and fourth picture only the nodes of the grid have been inverted and connected by straight lines. The third picture has been coloured in Photoshop by my wife Svetlana Morozova —the picture is included elsewhere in this MAPS— and is planned to become a stainless glass window, size 60x60cm, to decorate our house. The right picture has a small inversion circle; all the nodes of the grid except the central node are transformed to fit within the inversion circle.

Peaucellier-Lipkin Linkage is an intriguing mechanical device where to and fro motion along a straight line is transformed into a to and fro motion along a circular arc, or vice versa.



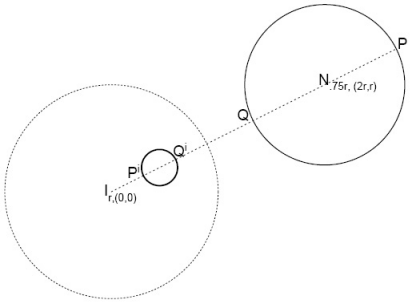
A disadvantage of this apparatus is that only a to and fro movement along a circular arc is obtained. Transformation of to and fro motion into rotation is realized in a car (think of pistons) via a crank-shaft or a totter system.

Each point of the circle is related to a point on the (right) line by inversion.[12]
In the device the point O and the centre of the circle are fixed. When the centre is connected to B, as in the animation, then the vertical movement of D results in movement of B along the circle. In the picture above (an arc of) the inversion circle, which transforms the circle into the straight line, is drawn. In a math animation we can just move D and calculate D^i and show these dynamically, no 'diamond' needed.

Circle inversion

The inversion of a circle not through the origin of an inversion circle is a circle. The inversion of a circle through the origin of an inversion circle is a straight line not through the origin, and vice versa. With the term generalized circle, to denote a line or circle, the above reads

... generalized circles invert into generalized circles...



The line from the centre I to the centre N intersects the circle $N_{.75r,(2r,r)}$ in two diametrical points. Inversion of these diametrical points yields diametrical points of the inverted circle.[13]

Properties

- Circle centres are collinear.
- Touching and intersection of generalized circles remain invariant.
- Circles orthogonal to the inversion circle remain invariant.
- Circles concentric with the inversion circle remain concentric with centres invariant.
- Two non-intersecting circles can be transformed into concentric circles.

The inverse of a circle $C_{r,(x,y)}$ with respect to the inversion circle $I_{k,(x_0,y_0)}$, is given by the circle $C^i_{r^i,(x^i,y^i)}$ with

$$\begin{aligned} x^i &= x_0 + s(x - x_0) \\ y^i &= y_0 + s(y - y_0) \quad \text{and} \quad s = \frac{k^2}{(x - x_0)^2 + (y - y_0)^2 - r^2} \\ r^i &= |s|r \end{aligned}$$

Special cases: the invariance of the inversion circle, and the invariance of the centres of the circles concentric with the inversion circle.

Calculation of the inverted circle

One can't simply invert the (centre of the) circle and the radius, because inversion does not preserve distances. Therefore use is made of two point inversions: inversion of diametrical points on the line through the centre of inversion and the circle centre.

PS code

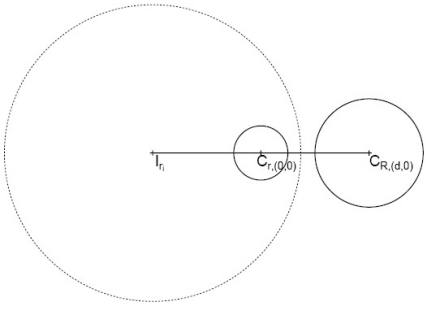
```
/circleinversion
% Nx, Ny, R: centre and radius of the to be inverted circle
% Ix, Iy, r: centre and  radius of the inversion circle
%==>
% xi,yi, ri: centre and radius of the inverted circle
{0 begin
/r exch def  /y exch def /x exch def
/R exch def  /Y exch def /X exch def
/Xmx X x sub def
/Ymy Y y sub def
%diametrical boundary points:  intersections of the line I--N with the to be
%                              inverted circle
/phi Ymy Xmx atan def
/bp1{/x1 phi cos R mul Xmx add def
     /y1 phi sin R mul Ymy add  def
     x1 y1}def
```

```
/bp2{/x2 Xmx phi cos R mul sub def
     /y2 Ymy phi sin R mul sub def
     x2 y2}def
bp1 0 0 r pointinversion
  /yi exch def /xi exch def
bp2 0 0 r pointinversion
  /py exch def /px exch def
  /yi py yi add .5 mul def
  /xi px xi add .5 mul def
/ri px xi sub py yi sub size def
/xi xi x add def /yi yi y add def % translate back
xi yi ri
end
} def
/circleinversion load 0 30 dict put
```

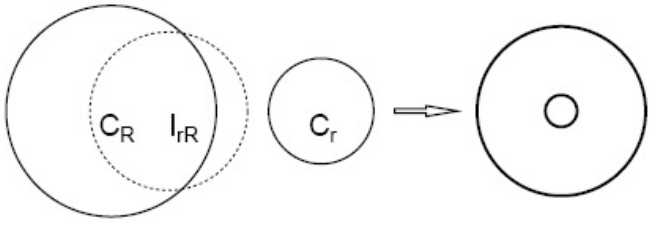
Applications

The property that inversion transforms generalized circles into generalized circles (and that inversion is conformal) makes it an extremely important tool of plane analytic geometry. By picking a suitable inversion circle, it is often possible to transform one geometric configuration into another, simpler one, in which a proof or calculation is more easily effected. http://mathworld.wolfram.com/Inversion.html.
Interesting! We'll see examples of this: in adding circles to Rerich's symbol, and in the solution of Apollonius problem by Circle Inversion.

The circle of anti-similitude also known as mid-circle, of two circles C and C^i is a circle for which C and C^i are inverses of each other. Suppose that the two circles are $C_{r,(0,0)}$ and $C^i_{R,(d,0)}$, then the centre of the inversion circle is $(-\mathrm{I}_x, 0)$, with $\mathrm{I}_x = d \cdot r/(R-r)$. The radius is $r_i = \sqrt{(\mathrm{I}_x + r) \cdot (\mathrm{I}_x + d - R)}$.



Any two circles can be made concentric by inversion by picking the inversion centre as one of the so-called limiting points. See http://mathworld.wolfram.com/LimitingPoint.html.



The dashed circle is the inversion circle I with abscissa of the centre $\mathrm{I}_{rR} = \frac{l \pm \sqrt{l^2 - 4d^2R^2}}{2d}$ with $l = d^2 - r^2 + R^2$. The main circle is $C_{R,(0,0)}$ and the small circle $C_{r,(d,0)}$. The radius of

the inversion circle determines the size of the concentric circles. See http://mathworld.wolfram.com/ConcentricCircles.html.

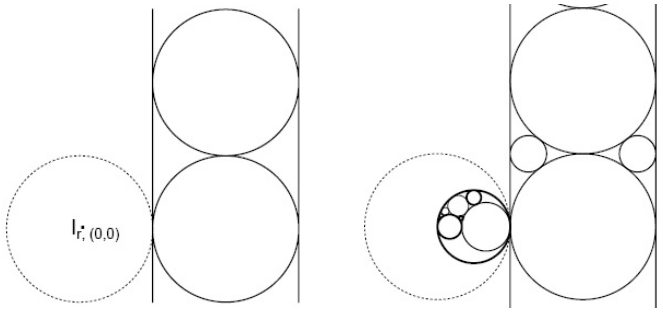
PS code The library operators can be found elsewhere in this note.

```
%!PS-Adobe-3.0 Two circles into concentric circles. CGL March 2010
%%BoundingBox: 0 0 620 790
(C:\\PSlib\\PSlib.eps) run %PS library

200 200 translate

/r 25 def /hr r 2 div def /2r r dup add def %radius of small circle
/R 50 def                                  %radius of big circle
/d 100 def     /2d d d add def             %distance between circle centres
/O {0 0} def
/Ix d d mul r r mul sub R R mul add
     d d mul r r mul sub R R mul add
     dup mul 2 d mul R mul dup mul sub sqrt
     sub
     2d div def % function of r and R

gsave
O moveto   -6 -12 rmoveto (C)
  H12pt setfont show 0 -3 rmoveto (R) H7pt setfont show
Ix 0 moveto 0 -12 rmoveto (I)
  H12pt setfont show 0 -3 rmoveto (rR) H7pt setfont show
d 0 moveto -6 -12 rmoveto (C)
  H12pt setfont show 0 -3 rmoveto (r) H7pt setfont show
grestore
/R1 R .75 mul def
gsave Ix 0 R1 0 360 invlin stroke grestore  %Inversion circle
gsave d 0 r 0 360 arc stroke grestore       %Cr
gsave 0 0 R 0 360 arc stroke grestore       %CR

135 0 translate
gsave newpath 0 0 30 0 2 5 15 arrow stroke grestore
30 0 translate

d 0 r   %small circle at (d,0)
Ix 0 R1 %inversion circle
circleinversion
/rinv exch def /yinv exch def /xinv exch def
xinv yinv rinv 0 360 arc stroke

0 0 R   %big circle at (0,0)
Ix 0 R1 %inversion circle
circleinversion
/rinv exch def /yinv exch def /xinv exch def
xinv yinv rinv 0 360 arc stroke
showpage
%%EOF
```

The article at http://www.partnership.mmu.ac.uk/cme/Geometry/CoaxalInvers/InversionCoaxalCircles.html treats the problem of converting two circles into concentric circles by circle inversion in Java. The concept of radical axis of two circles is used. Another reference is http://www.cut-the-knot.org/ctk/Circle.shtml, also with animation.

Circle covered by touching circles

Let us consider three circles and two verticals as given in the picture below at the left.

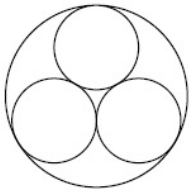


If we invert the stacked circles and bordering verticals in I_r, see the left picture, then we arrive at the picture at the right. Well ... not quite: two small circles have been added. In the right picture the left vertical is converted into 'the main' circle which is covered by the other inverted circles.

Remark In the mid-nineties I drew the above picture after having solved in PostScript the equations for the (inverted) touching circles, without the use of Circle Inversion.[14]

Rerich Trinity symbol

Н К Рерих[15] painted the Trinity symbol, well ... sort of.



$r = \langle parameter \rangle \; x = 0 \; y = r_i + r$ (upper inner circle)

$r_i = \dfrac{2-\sqrt{3}}{\sqrt{3}} r \qquad x = 0 \; y = 0$ (inscribed circle, not shown)

$R = \dfrac{2+\sqrt{3}}{\sqrt{3}} r \qquad x = 0 \; y = 0$ (circumscribed circle)

What is the inversion of this symbol? How can we cover the (circumscribed) circle with smaller circles, fractal-like?

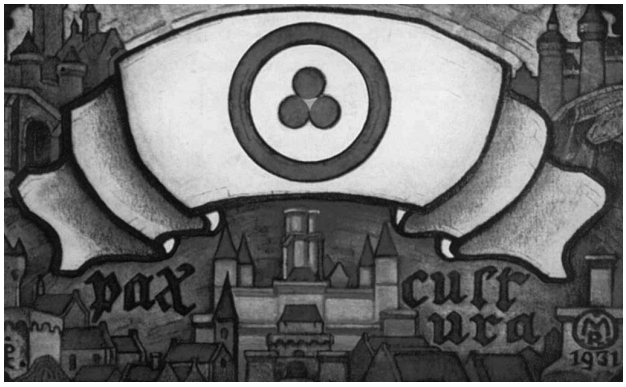
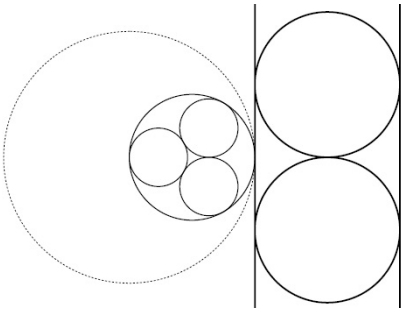
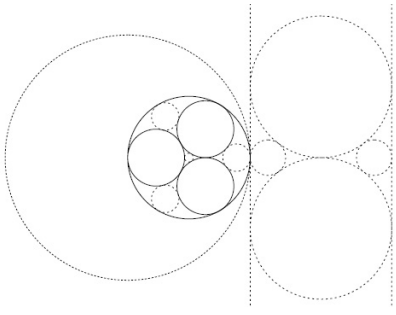


Figure 1. Rerich's *Pax Cultura*

By inversion Invert the picture in a circle with as radius the diameter of the circumscribing circle and a point on the circumference of the main circumscribing circle as inversion point.



The picture is rotated 90 degrees counterclockwise for convenience. The right stacked circles are the inversions of two of Rerich's inner circles C_r. The inversion circle is dashed. The verticals are the inversion of the main circumscribing circle and the Rerich inner circle which passes through the inversion centre.

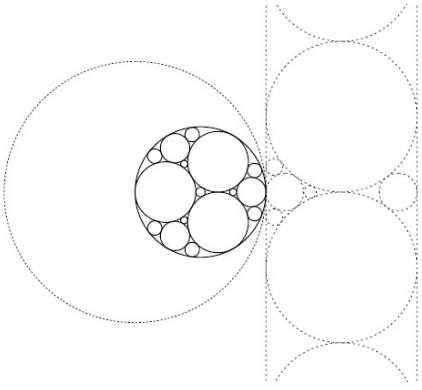


In the transformed figure the radius (and centre) of the small circle can be obtained as .25r^i, where r^i is the radius of the inverted circles C_r^i. The middle circle is added in the Trinity symbol, and the second level Repux is obtained via backtransformation and a rotation or two in the original picture, as can be read from the code below. The library operators can be found elsewhere in this note.

```
%!PS Rerich 2nd level Circles. CGL March 2010
%%BoundingBox: 0 0 620 790
(C:\\PSlib\\PSlib.eps) run                 % the library

200 300 translate

90 rotate
/r 25 def /yr r 3 sqrt div dup add def  /xr 0 def /2r r r add def
/R yr r add def /mR R neg def /2R R R add def /m2R 2R neg def
/3R 2R R add def /m3R 3R neg def
/lw .1 def    lw setlinewidth

0 0 R 0 360 arc stroke                     % Circumscribing circle
 3{newpath xr yr r 0 360 arc stroke        % Rerich circles
    120 rotate}repeat

gsave
0 R  2R 0 360 arc                          % inversion circle
invcir  stroke
grestore

Auxlin
m3R mR moveto 3R mR lineto stroke          % 'Inverted' circle is vertical
0 R 2r sub                                 % point to be inverted.  Actually
                                           % the inversion of the circle
```

```
0 R 2R pointinversion /yi exch def /xi exch def
m3R yi moveto 3R yi lineto stroke          % 'Inverted' circle is vertical

xr yr -120 rot /yrrot exch def /xrrot exch def
xrrot yrrot r                              % to be inverted Rerich circle
0 R 2R circleinversion /rinv exch def /yinv exch def /xinv exch def
xinv yinv rinv 0 360 arc stroke            % Inverted Rerich circle
%
xrrot neg yrrot r                          % to be inverted Rerich circle
0 R 2R circleinversion /rinv exch def /yinv exch def /xinv exch def
xinv yinv rinv 0 360 arc stroke            % Inverted Rerich circle
/smallcircle {0 mR rinv -4 div add  rinv 4 div} def
                                           % mnemonic for centre and radius
smallcircle 0 360 arc stroke
smallcircle 0 R 2R circleinversion
  /rsmall exch def /ysmall exch def /xsmall exch def

xsmall ysmall rsmall 0 360 arc stroke      % Snd level Rerich circle
xsmall ysmall 120 rot /ysmall exch def /xsmall exch def
xsmall ysmall rsmall 0 360 arc stroke      % Snd level Rerich circle
xsmall neg ysmall rsmall 0 360 arc stroke  % Snd level Rerich circle
% middle circle
/rmid R 2r sub def /xmid 0 def /ymid 0 def
xmid ymid rmid 0 360 arc stroke
xmid ymid rmid 0 R 2R circleinversion
  /rmidinv exch def /ymidinv exch def /xmidinv exch def
xmidinv ymidinv rmidinv 0 360 arc Auxlin stroke
%
showpage
```

Continuation to the third level, where touching circles in the transformed picture can be calculated by solving a quadratic equation in one unknown, yields



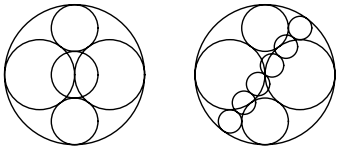
By Soddy's formula which gives a relation between the radii of touching circles, and goes back to Descartes' circle theorem

$$2\sum_{i=1}^{4}\frac{1}{r_i^2}=\left(\sum_{i=1}^{4}\frac{1}{r_i}\right)^2$$

which is of use in the so-called four coins problem. For continuation to the limit Soddy's formulas might be used. The formula will be given later.

In general a numerical analysis approach, where one has to solve 1 quadratic equation with nested a system of 2 linear equations in 2 unknowns, is less cumbersome than the Circle Inversion method.

Sangaku In Pythagoras,[16] april 2010, Bernard Asselbergs mentions the Japanese sangaku, which are diagrams to illustrate mathematical properties. He also derives Soddy's formula, the relation between the radii of mutual tangent circles, from the equations of the Heron formula $-\sqrt{s(s-a)(s-b)(s-c)}$, with a, b, c the sides and s half the sum of the sides— for the surface of triangles, which are obtained by connecting the centres of the mutual tangent circles.



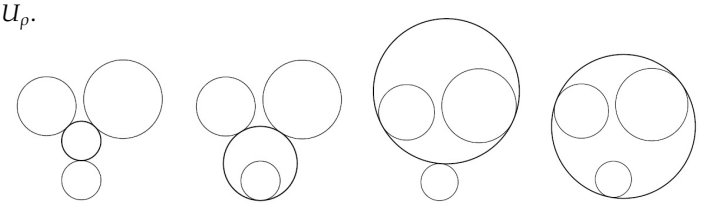
Interesting visual result of some relations between the radii of mutual tangent circles.

Apollonius problem

The problem how to draw touching circles to three arbitrary circles is named after Apollonius and can be solved analytically by the inversion technique.[17]
There are 8 possible solutions:

1. the unknown circle inscribes the three circles (left figure below (1 way))
2. the unknown circle circumscribes the three circles (fourth figure below (1 way))
3. the unknown circle surrounds one circle (second figure below (3 ways))
4. the unknown circle surrounds two circles (third figure below (3 ways))

Let us denote the three given circles by A, B, C, and the sought after, unknown tangent circle by U_ρ.



Solutions to Apollonius' problem are generally considered in pairs; for each solution circle, there is a conjugate solution circle. One solution circle excludes the given circles that are enclosed by its conjugate solution, and vice versa. The conjugate solution circles are related by inversion, with the so-called radical circle, which is perpendicular to the three given circles, as inversion circle. The inscribed and circumscribed circles form such a pair. The solution circles as given in the 2nd and 3rd picture above also form a conjugate pair.

Calculation 1st and 2nd case: unknown circle is surrounded by the three circles: the inscribed circle, and the unknown circle surrounds all, the circumscribed circle See the first phase in Figure 2 for the three given circles A, B, and C.
First step. Let us increase the radii of the three given circles by d (as a consequence the radius of the unknown circle U is decreased with the same amount in the transformed situation) such that 2 circles out of A, B, and C will touch each other.[18] Let us choose this tangent point as the centre of the inversion circle, and call it I. See the second phase in Figure 2.
Second step. Invert the circles A, B, and C in the circle with centre I and radius the diameter of the (enlarged) C. The (enlarged) circles B and C become the parallel lines B^i and C^i, while the (enlarged) circle A becomes the circle A^i, drawn dashed. See the right part in Figure 2. Construct the circle which touches (the circle) A^i, (and the lines)

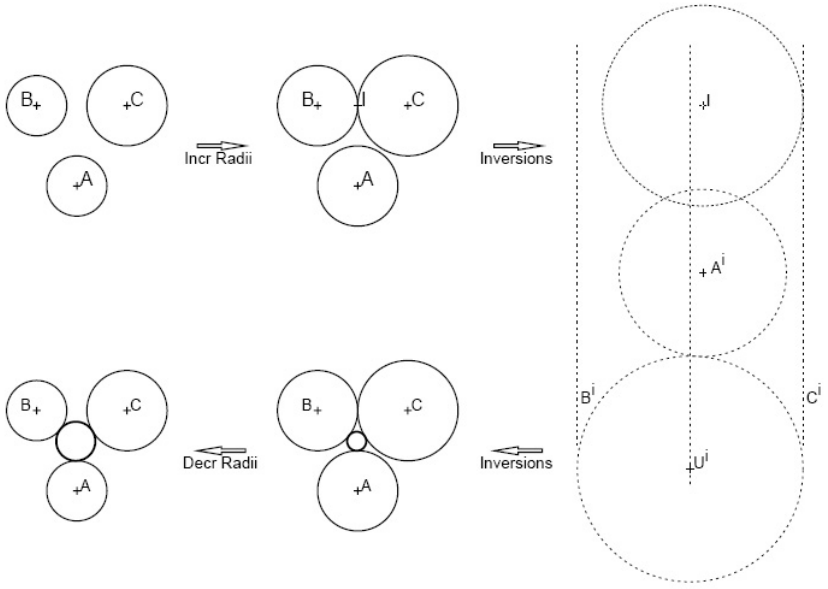


Figure 2. Inscribed circle

B^i, C^i, this is U^i, drawn dotted.[19] Its radius r is half the distance between B^i and C^i. Its centre is the intersection point of the line midway between B^i and C^i and the circle about the centre A^i with radius $r_{A^i} + r$.
Finally, by drawing the inverse of U^i we have found the centre of the required (inside) Apollonius circle U. See low middle part of Figure 2. Correct for the radius and the solution is given left below in Figure 2, bold.

The circumscribed circle: 2nd case In the above description of the algorithm we have neglected the other intersection point of the line midway between B^i and C^i and the circle about the centre A^i with radius $r_{A^i} + r$. This intersection point is the centre of the circumscribing circle, U^i_{cir}, the 2nd case.

Without further ado, I have drawn this circle in the lower part of Figure 3.

The unknown circle envelopes one circle and touches the other two circles on the outside: 4th case The algorithm is similar to the above, but varies in details. Let us assume that circle A will be enveloped by the unknown circle. If we increase the radii of B and C such that they touch, the radius of the circle A has to be *decreased*. Moreover, the circle U^i has to touch the circle A^i on the other side, compare Figure 2 and Figure 4. Details!

The unknown circle envelopes two circles and touches the third circle on the outside: 3rd case Let us concentrate on the case when A will be touched on the outside and B and C will be contained in the enveloping (unknown) circle. The algorithm must be adjusted when we increase the radii of B and C (and therefore also the radius of the unknown circle U) by decreasing the radius of A, and then perform the algorithm.

For more illustrations see http://mathforum.org/mathimages/index.php/Problem_of_Apollonius for example.
See http://en.wikipedia.org/wiki/Circles_of_Apollonius for a coloured version of the all-in-one picture.

Once we have written the PostScript program we can abstract into an operator, where the three circles are provided on the stack, and we'll find the touching circles, inner and outer, after completion on the stack.

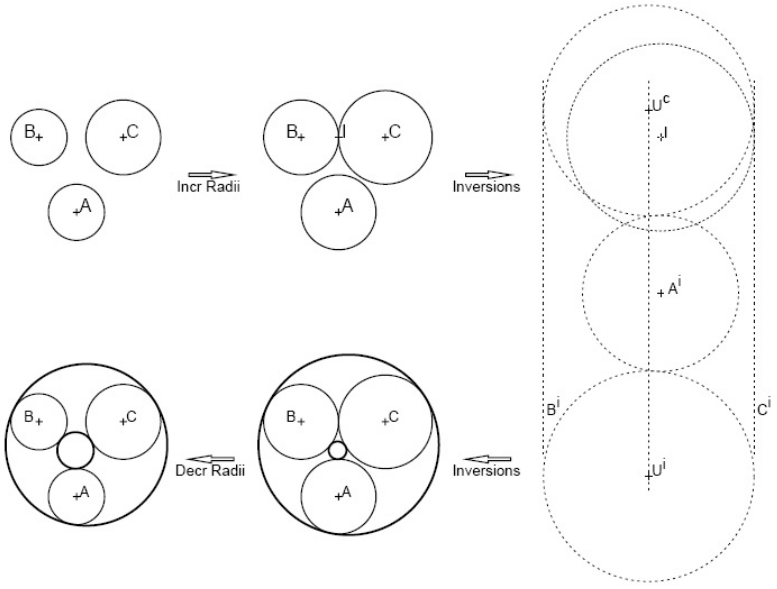


Figure 3. In- and circumscribed circle

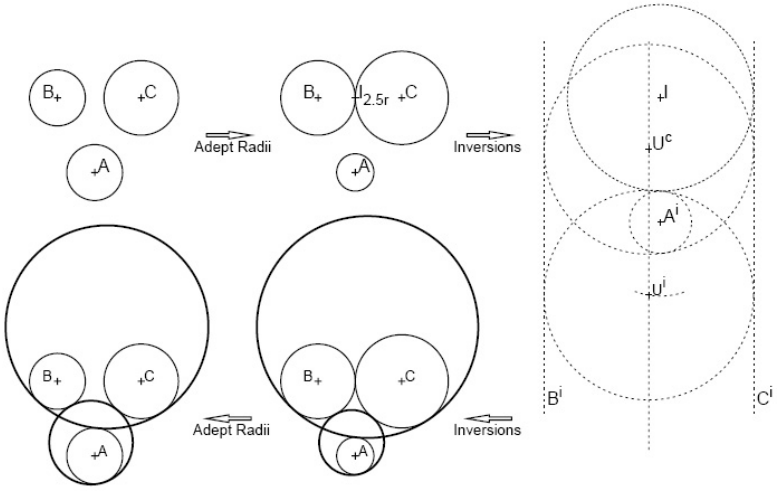


Figure 4. Two circles inside and one outside and vice versa

The previous requires work to get all the details right. I did not pursue this route via the Circle Inversion technique. However … the numerical method, to be discussed later in this note, is less cumbersome, and will yield the operator `Apollonius` for this purpose.

What if one circle contains the two other circles? Then the solutions touch the enveloping circle on the inside and the other circles on the outside.
See http://mathworld.wolfram.com/ApolloniusProblem.html for a thorough discussion of Apollonius problem and its solutions.

The numerical method to be introduced later in this note will yield the operator `Apollonius2` suited for this case.

Solution of Apollonius problem by numerical analysis techniques

The conditions for the inscribed circle, $C_{r,(x,y)}$, of three (distinct) circles, $C^k, k = 1,2,3$, read

$$\|C^k - C_{r,(x,y)}\| = r_k + r, \quad k = 1,2,3$$

three quadratic equations in three unknowns, rather complex.

But ... we can simplify.

If we square the conditions and subtract 2 from 1 and 3 from 2, we arrive at

$$\begin{pmatrix} x_{21} & y_{21} \\ x_{32} & y_{32} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \mathrm{f}_r(21) \\ \mathrm{f}_r(32) \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} r + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

meaning x and y are linear in r. Together with one of the squared conditions

$$\mathrm{H}_{xy}(r) = 0$$

we have transformed the problem into 2 linear equations and one quadratic in r.

Bounds for r are[20]

$$\begin{aligned} r &\geq \min_{m,n}(\|C^m - C^n\| - r_m - r_n) \\ r &\leq \max_{m,n}(r_m + r_n) \end{aligned} \qquad m,n = 1,2,3$$

$\mathrm{H}_{xy}(r) = 0$, I solved ≈15years ago by bisection programmed in PostScript. I'll rehearse on it later in this note with programs in PostScript and MetaPost. Moreover ... I looked closer at the quadratic equation $\mathrm{H}(r) = 0$ and arrived at the simplest problem variant, which I did not realize ≈15 years ago. We'll come back on it in this note.

Throughout linear system? If we subtract the squared conditions from each other then we arrive at the linear equations

$$\begin{pmatrix} x_{21} & y_{21} & r_{21} \\ x_{32} & y_{32} & r_{32} \\ x_{31} & y_{31} & r_{31} \end{pmatrix}\begin{pmatrix} x \\ y \\ r \end{pmatrix} = \begin{pmatrix} \mathrm{g}_{21} \\ \mathrm{g}_{32} \\ \mathrm{g}_{31} \end{pmatrix}$$

with

$$\begin{aligned} z_{ij} &= z_i - z_j \\ \bar{z}_{ij} &= \frac{z_i + z_j}{2} \\ \mathrm{g}_{ij} &= x_{ij}\bar{x}_{ij} + y_{ij}\bar{y}_{ij} - r_{ij}\bar{r}_{ij} \end{aligned} \qquad z = x,y,r$$

But ... too nice to be true?[21] Let us pursue it nevertheless and see what we'll stumble upon.

Throughout linear system: in MetaPost

```
beginfig(0);
r1=50;      x1=0;               y1=-2r1;
r2= 1.5r1;  x2=-(r1+r2)/sqrt2;  y2=-x2;
r3=   2r1;  x3= (r1+r3)/sqrt2;  y3=x3;
path p; p:=fullcircle scaled 2;
draw p scaled(r1) shifted(x1,y1);%...
%data
x21=x2-x1; ... mx21=(x2+x1)/2; ...
%equations
x21 * x + y21 * y + r21 * r = x21*mx21+y21*my21-r21*mr21;
x32 * x + y32 * y + r32 * r = x32*mx32+y32*my32-r32*mr32;
x31 * x + y31 * y + r31 * r = x31*mx31+y31*my31-r31*mr31;
```

```
...
endfig
```

Which yields ...

```
... inconsistent equation (off by 0.00356)
```

l28... $x31*x+y31*y+r31*r = x31*mx31+y31*my31-r31*mr31$; Meaning ... singular system! Neat of MetaPost to warn in this way.

Throughout linear system: in PostScript

```
%!PS Solve 33 linear system. cgl March 2010
%data
/x21 x2 x1 sub def  ...
/y21 y2 y1 sub def ...
/r21 r2 r1 sub def   ...
/mx21 x2 x1 add 2 div def  ...
/my21 y2 y1 add 2 div def ...
/mr21 r2 r1 add 2 div def   ...
/rh1 x21 mx21 mul  y21 my21 mul add
      r21 mr21 mul sub def ...
%solve equations
rh1 x21 y21 r21
rh2 x32 y32 r32
rh3 x31 y31 r31
           solve33
          /rxy exch def /y exch def /x exch def
250 300 translate
x1 y1 r1 0  360 arc blue stroke...
x y rxy  0 360 arc blue [1] 0 setdash stroke
showpage
```

I was surprised by the result: the picture showed up?!?

Note that I warn the user by giving the value of the determinant of the matrix, 0.0625 for this case. A singular system! For the occasion I wrote the PS operator `solve33`, which uses partial pivoting and invokes `solve22`. See Appendix I.

Inscribed circle in PostScript Like ≈15 years ago, I solved

$$\mathrm{H}_{x,y}(r) = 0$$

and

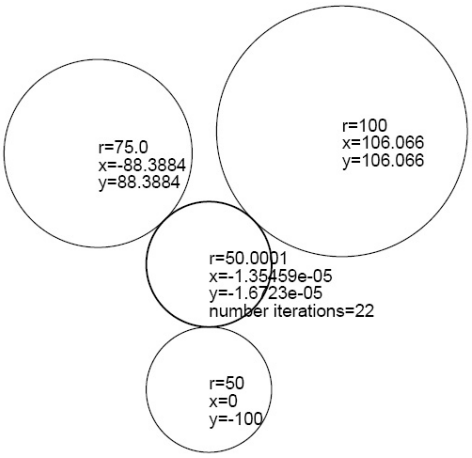
$$\begin{pmatrix} x_{21} & y_{21} \\ x_{32} & y_{32} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \mathrm{f}_r(21) \\ \mathrm{f}_r(32) \end{pmatrix}$$

where

$$\mathrm{H}_{x,y}(r) = r + r_3 - \|C^3 - C_{r,(x,y)}\|$$

The computation scheme reads
- start with estimate of r
- calculate (x, y) by solving the 2x2 linear system of equations
- calculate $\mathrm{H}_{x,y}(r)$ and compare this value with the value of H at the interval ends
- adjust the interval by increasing the lower value or decreasing the upper value of the interval for r
- take the mean of the interval ends as new estimate of r
- repeat the above process, until either the size of the interval is small enough or H is close enough to zero.

The result can be seen in the next included picture.



As extra I have included the number of iterations, 22 for this case. A more sophisticated zero-finding operator, with super-linear convergence, could have been programmed; the simple bisection algorithm, with linear convergence, is good enough for the occasion.[22]

```
%!PS Inscribed circle to three circles  BachoTeX2010 cgl March 2010
%%BoundingBox: 0 0 620 790
(C:\\PSlib\\PSlib.eps) run          %PS library
/nstr 15 string def
% circles
/r 50 def
/r1 r def          /x1 0 def                        /y1 r r1 add neg def
/r2 r 1.5 mul def  /x2 r r2 add sqrt2 div neg def   /y2 x2 neg def
/r3 r 2 mul def    /x3 r r3 add sqrt2 div def       /y3 x3 def
% auxiliaries
/x21 x2 x1 sub def  /x31 x3 x1 sub def  /x32 x3 x2 sub def
/y21 y2 y1 sub def  /y31 y3 y1 sub def  /y32 y3 y2 sub def
/r21 r2 r1 sub def  /r31 r3 r1 sub def  /r32 r3 r2 sub def
/mx21 x2 x1 add 2 div def /mx31 x3 x1 add 2 div def /mx32 x3 x2 add 2 div def
/my21 y2 y1 add 2 div def /my31 y3 y1 add 2 div def /my32 y3 y2 add 2 div def
/mr21 r2 r1 add 2 div def /mr31 r3 r1 add 2 div def /mr32 r3 r2 add 2 div def
% right-hand side as function of r for inscribed circle
/rh1 {x21 mx21 mul  y21 my21 mul add r21 mr21 mul sub r21 r mul sub} def
/rh2 {x32 mx32 mul  y32 my32 mul add r32 mr32 mul sub r32 r mul sub} def
% bounds for r
/d21 x21 y21 size def
/d32 x32 y32 size def
/d31 x31 y31 size def
/dr21 d21 r2 sub r1 sub def% function for inscribed circle
/dr32 d21 r3 sub r2 sub def% function for inscribed circle
/dr31 d31 r3 sub r1 sub def% function for inscribed circle
dr21 0 lt {(Circle s 1 2 intersect) print} if
dr32 0 lt {(Circle s 3 2 intersect) print} if
dr31 0 lt {(Circle s 3 1 intersect) print} if
/rmin dr21 def
dr32 rmin lt {/rmin dr32 def}if
dr31 rmin lt {/rmin dr31 def}if
/rmax d21 def
```

```
d32 rmax gt {/rmax d32 def}if
d31 rmax gt {/rmax d31 def}if

/cnt 0 def
/eps 0.01 def % required accuracy
/nmax 50 def% maximum number of iterations

/d3xy {x3 x sub y3 y sub size} def% distance circle 3 to estimated circle
/H {d3xy r3 sub r sub} def        % function for inscribed circle

% Calculation values of H(r) in endpoints for r. Interval is [rmin, rmax]
/r rmin def
rh1 x21 y21
rh2 x32 y32 solve22
  /y exch def /x exch def %/d exch def % not yet in solve22
/Hrmin H def
/r rmax def
rh1 x21 y21
rh2 x32 y32 solve22
  /y exch def /x exch def %/d exch def % not yet in solve22
/Hrmax H def
Hrmax Hrmin mul
  0 gt {(error no opposite signs: Hrmin X Hrmax > 0) print} if

% bisection zerofinding
nmax{/cnt cnt 1 add def
   rmax rmin sub abs  eps gt
   Hrmax abs eps 100 div gt or
   Hrmin  abs eps 100 div gt or
   {/r rmin rmax add 2 div def % bisection
    rh1 x21 y21
    rh2 x32 y32 solve22
            /y exch def /x exch def %/d exch def
    /Hr H def
    Hr Hrmax mul 0 gt {/Hrmax Hr def /rmax r def}
                              {/Hrmin Hr def /rmin r def} ifelse
    }{exit}ifelse
}repeat % or say loop for infinite case, but to limit it is safer
/rxy rmin rmax add 2 div def

250 300 translate

gsave
x y rxy  0 360 arc stroke % the looked for inscribed circle
0 0 moveto (r=) H10pt setfont show r nstr cvs show
0 -12 moveto (x=)  show x nstr cvs show
0 -24 moveto (y=)  show y nstr cvs show
0 -36 moveto (number iterations=)  show cnt nstr cvs show
0 -48 moveto (determinant=)  show d nstr cvs show
grestore

newpath x1 y1 r1 0  360 arc  stroke % first original circle
x1 y1 moveto (r=) H10pt setfont show r1 nstr cvs show
x1 y1 12 sub moveto (x=)  show x1 nstr cvs show
x1 y1 24 sub moveto (y=)  show y1 nstr cvs show

newpath x2 y2 r2 0  360 arc stroke % second original circle
x2 y2 moveto (r=) H10pt setfont show r2 nstr cvs show
```

```
x2 y2 12 sub moveto (x=)  show x2 nstr cvs show
x2 y2 24 sub moveto (y=)  show y2 nstr cvs show

newpath x3 y3 r3 0  360 arc  stroke 5third original
x3 y3 moveto (r=) H10pt setfont show r3 nstr cvs show
x3 y3 12 sub moveto (x=)  show x3 nstr cvs show
x3 y3 24 sub moveto (y=)  show y3 nstr cvs show
```

Inscribed circle in MetaPost When coding along the same lines as in PostScript, but with using MetaPost's equation solving possibilities, one has to be aware that equations within a loop must be specified with arrays for the unknowns.

```
if scantokens(mpversion) > 1.005:
  outputtemplate :=
else:
  filenametemplate
fi
"%j.eps";
beginfig(0);
numeric x[],y[]; %showdependencies; tracingequations:=1 ;
r1=50;      x1=0;                  y1=-2r1;
r2= 1.5r1; x2=-(r1+r2)/sqrt2; y2=-x2;
r3=   2r1; x3=  (r1+r3)/sqrt2; y3=x3;
path p; p:=fullcircle scaled 2;%more convenient, because the diameter is the unit
drawoptions(withcolor blue);
draw p scaled(r1) shifted(x1,y1);
draw p scaled(r2) shifted(x2,y2);
draw p scaled(r3) shifted(x3,y3);

x21=x2-x1; x32=x3-x2; x31=x3-x1;
y21=y2-y1; y32=y3-y2; y31=y3-y1;
r21=r2-r1; r32=r3-r2;  r31=r3-r1;
mx21=(x2+x1)/2; mx32=(x3+x2)/2;  mx31=(x3+x1)/2;
my21=(y2+y1)/2;my32=(y3+y2)/2;  my31=(y3+y1)/2;
mr21=(r2+r1)/2;  mr32 =(r3+r2)/2;  mr31=(r3+r1)/2;
rmin=25; rmax=100;
%r:= rmin;
rh1:= x21*mx21+ y21*my21-r21*mr21-r21*rmin;
rh2:= x32*mx32+y32*my32-r32*mr32-r32*rmin;
%solve 22 with rmin in rh1 rh2; MP's way
x21 * xmin + y21 * ymin = rh1;
x32 * xmin + y32 * ymin = rh2;
Hrmin :=  abs(x3 -xmin, y3-ymin)-r3 -rmin;
%r:= rmax;
rh1:= x21*mx21+y21*my21-r21*mr21-r21*rmax;
rh2:= x32*mx32+y32*my32-r32*mr32-r32*rmax;
%solve 22 with rmax  in rh1 rh2; MP's way
x21 * xmax + y21 * ymax = rh1;
x32 * xmax + y32 * ymax = rh2;
Hrmax :=  abs(x3 -xmax, y3-ymax)-r3 -rmax;

nmax:=50; eps:=0.01;
for i:=1 upto nmax: cnt:=i;
exitif (rmax -rmin)< eps ;
r := (rmin + rmax)/2;
rh1:= x21*mx21+ y21*my21-r21*mr21-r21*r;
rh2:= x32*mx32+y32*my32-r32*mr32-r32*r;
```

```
x21 * x.i + y21 * y.i = rh1;
x32 * x.i + y32 * y.i = rh2;
Hr:= abs(x3 -x.i, y3-y.i)-r3 -r;
if Hr*Hrmax > 0:
Hrmax:=Hr; rmax:=r;
else:
Hrmin:=Hr;rmin:=r;
fi;
endfor
%showvariable rmin, Hrmin, rmax, Hrmax;
draw p scaled(r) shifted(x.i, y.i) withcolor red;
currentpicture:=currentpicture shifted(200,300);
endfig;
end
```

Correct results were obtained.

Circumscribed circle in PostScript The conditions for the circumscribed circle are

$$\{r,(x,y) \mid \|C^k - C_{r,(x,y)}\| = r - r_k, \quad k = 1,2,3\}$$

Squaring the conditions and subtracting 2 from 1 and 3 from 2, yields the linear system[23]

$$\begin{pmatrix} x_{21} & y_{21} \\ x_{32} & y_{32} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f_r(21) \\ f_r(32) \end{pmatrix}.$$

Together with e.g. the 3rd original condition

$$\mathrm{H}_{x,y}(r) = r - r_3 - \|C^3 - C_{r,(x,y)}\| = 0$$

we arrive at three equations of which one is quadratic and two are linear.

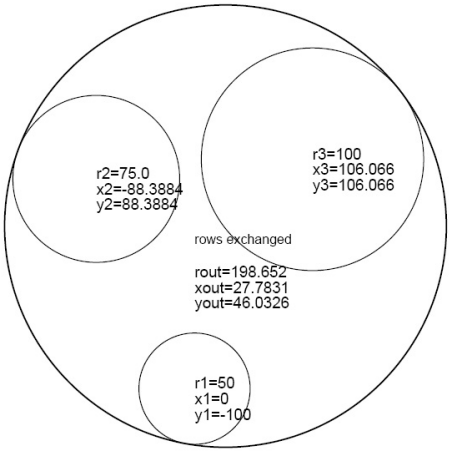
Auxiliaries

$$\mathrm{f}_r(ij) = x_{ij}\bar{x}_{ij} + y_{ij}\bar{y}_{ij} - r_{ij}\bar{r}_{ij} + r_{ij}r$$

bounds for r

$$\begin{aligned} r &\geq \min_{m,n}(\|C^m - C^n\| - r_m - r_n) \\ r &\leq \max_{m,n}(\|C^m - C^n\| \end{aligned} \qquad m,n = 1,2,3.$$

Modifying the program towards the circumscribing circle case,[24] yields as results



The number of iterations is 20.

Circumscribed circle in MetaPost I stumbled upon the limitations of the number system of the current MetaPost: overflow occurred. I had to scale the problem, after which the correct results were obtained.

Mutual tangent circles: Soddy's formula In the mid-nineties J.H. van de Stadt communicated Soddy's formula to me, which is an explicit solution of H(*r*) = 0.
The radius, r, of the inscribed circle is given by

$$\frac{1}{r} = \frac{1}{r_a} + \frac{1}{r_b} + \frac{1}{r_c} + 2\sqrt{\frac{1}{r_a r_b} + \frac{1}{r_b r_c} + \frac{1}{r_c r_a}}$$

Similarly, the radius, R, of the circumscribed circle is given by[25]

$$\frac{1}{R} = -\frac{1}{r_a} - \frac{1}{r_b} - \frac{1}{r_c} + 2\sqrt{\frac{1}{r_a r_b} + \frac{1}{r_b r_c} + \frac{1}{r_c r_a}}$$

These formulas can also be used for the case when the outside (or inside) circle is known and one of the other circles has to be determined.

Soddy's formula goes back to Descartes circle theorem, which for the 4 circles reads

$$2\sum_{i=1}^{4} \frac{1}{r_i^2} = \left(\sum_{i=1}^{4} \frac{1}{r_i}\right)^2$$

I did not make use of these beautiful results, because there is more to it then just the radius. Moreover, in Apollonius problem the given circles don't have to touch each other.

The Solution of Apollonius problem: operator Apollonius

As earlier, squaring the conditions and subtracting 2 from 1 and 3 from 2, gives 2 linear equations with three unkowns. We'll pursue this for the inscribed circle, and we'll see that all cases are solved by the resulting operator `Apollonius`.

If we express in the linear subsystem x and y in r, we arrive at

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} r + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Substitution of the above in the squared 3rd condition

$$(x - x_3)^2 + (y - y_3)^2 = (r + r_3)^2$$

yields one quadratic equation in one unknown

$$A \cdot r^2 - 2B \cdot r + C = 0$$

with the explicit solutions

$$r_{1,2} = \frac{B}{A} \pm \sqrt{(\frac{B}{A})^2 - \frac{C}{A}}$$

The various constants can be read from the PostScript snippet below, were x32 denote the difference between x3 and x2, and mx32 denote the mean of x3 and x2 etc.

Only one solution is real for the case of distinct circles, that with the + sign, the other, the spurious solution, has sneaked in while squaring the third condition.

```
%solve (in the real code I exchange rows if necessary)
% / x21 y21 \   / x \       / g21 \       / a1 \          / b1 \
% |          | |     |  =  |       | =   |      |  r +   |      |
% \ x32 y32 /   \ y /       \ g32 /       \ a2 /          \ b2 /
/p  x32 x21 div def% pivot
/a22 y32 p y21 mul sub def
```

```
/a2 r32 neg p r21 mul add a22 div def
/b2 g32 p g21 mul sub a22 div def
/a1 a2 y21 mul r21 add neg x21 div def
/b1 g21 y21 b2 mul sub x21 div def
%express  x and y as function of r
/x {a1 r mul b1 add} def%linear in r
/y {a2 r mul b2 add} def%linear in r
%Coefficients of quadratic equation A*r^2 - 2B*r + C = 0
/A a1 dup mul a2 dup mul add 1 sub def
/B r3 a1 b1 x3 sub mul sub
       a2 b2 y3 sub mul sub  def
/C b1 x3 sub dup mul b2 y3 sub dup mul add r3 dup mul sub def
%the radius of the inscribed circle (neglecting the A=0 case here)
/r B A div dup dup mul C A div sub sqrt add def
%draw inscribed circle, in red and dotted
x y r  0 360 arc [1] 0 setdash red stroke
```

Neat!

Although not really higher mathematics, I did not find this last result in Courant & Robbins, nor in the numerical Math books I own, nor did I come across it in my early num math career. In the enormous useful *wikipedia* encyclopaedia http://en.wikipedia.org/wiki/Problem_of_Apollonius the history of the problem and a kaleidoscopic survey of solution techniques are presented, included the one treated above.

Use of the `inscribed` operator, where x_i y_i r_i denote the centre and radius of the calculated inscribed circle, is done by the invoke

```
x1 y1 r1 x2 y2 r2 x3 y3 r3  inscribed /ri exch def /yi exch def /xi exch def
```

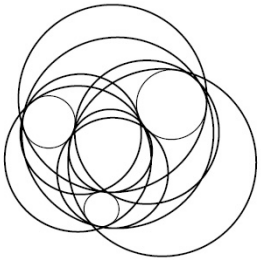
The operator `circumscribed` is highly similar. For the invoke change `inscribed` by `circumscribed` in the example line given above.

The pearl of this paper is the unifying operator `Apollonius`, which can give all 8 solutions

- the inscribed circle, with an invoke similar to the above with `inscribed` changed by `Apollonius`
- the circumscribed circle, with the invoke with the negative radii

```
x1 y1 r1 neg
x2 y2 r2 neg
x3 y3 r3 neg Apollonius
             /rcircum exch def /ycircum exch def /xcircum exch def
```

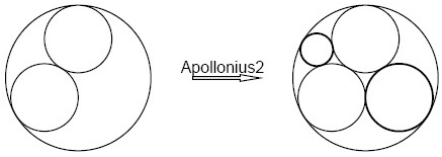
- the other cases, obtained by the invoke with appropriate positive and negative radii.



Not so clear in B&W, later in this note I'll disentangle the picture.

Two circles inside one circle

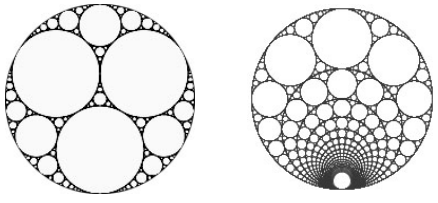
So far I qualified the second root as a spurious solution of the quadratic equation in r. In the case of this section we do have two solutions, so the second value for r, together with its centre, are also delivered by `Apollonius2`.



The invoke is similar and for the Rerich case, when we like to draw the second order circles, the invoke reads

```
newpath 0    0  R 0 360 arc stroke
newpath x1  y1 r1 0 360 arc stroke
newpath x2  y2 r2 0 360 arc stroke
x1 y1 r
x2 y2 r
0   0  R neg   Apollonius2  /rsnd1 exch def /ysnd1 exch def /xsnd1 exch def
                           /rsnd2 exch def /ysnd2 exch def /xsnd2 exch def
green %or a setdash when in b&w
newpath xsnd1 ysnd1 rsnd1 0 360 arc stroke
newpath xsnd2 ysnd2 rsnd2 0 360 arc stroke
```

Beautiful Apollonius gaskets I borrowed from the WWW If we start with one circle and within this circle a series of circles which touch each other, then one may obtain



Interesting theorems exist, for example Steiner's alternative, about circles covered by touching circles. http://www.cgl.ucsf.edu/home/bic/steiner/asilomar_2005_steiner_5a .ppt. I'm not aware of the usefulness of Steiner's l'art pour l'art alternative.
The left figure was already known in Japan as a sangaku in 1788.

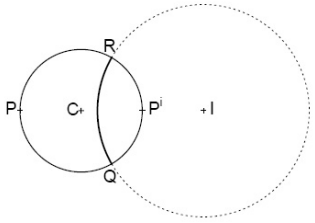
Applications [26]

The principal application of Apollonius' problem, as formulated by Isaac Newton, is hyperbolic trilateration, which seeks to determine a position from the differences in distances to at least three points. For example, a ship may seek to determine its position from the differences in arrival times of signals from three synchronized transmitters. Solutions to Apollonius' problem were used in World War I to determine the location of an artillery piece from the time a gunshot was heard at three different positions. Hyperbolic trilateration is the principle used by the Decca Navigator System and LORAN. Similarly, the location of an aircraft maybe determined from the difference in arrival times of its transponder signal at four receiving stations. This multilateration problem is equivalent to the three dimensional generalization of Apollonius' problem and applies to global positioning systems such as GPS. It is also used to determine the position of calling animals (such as birds and whales), although Apollonius' problem does not pertain if the speed of sound varies with direction (i.e., the transmission medium is not isotropic).

Apollonius' problem has other applications. In Book 1, Proposition 21 in his Principia, Isaac Newton used his solution of Apollonius' problem to construct an orbit in celestial mechanics from the centre of attraction and observations of tangent lines to the orbit corresponding to instantaneous velocity. The special case of the problem of Apollonius when all three circles are tangent, Rerich's symbol, is used in the Hardy-Littlewood circle method of analytic number theory to construct Hans Rademacher's contour for complex integration, given by the boundaries of an infinite set of Ford circles each of which touches several others. Finally, Apollonius' problem has been applied to some types of packing problems, which arise in disparate fields such as the error-correcting codes used on DVDs and the design of pharmaceuticals that bind in a particular enzyme of a pathogenic bacterium.

Orthogonal circles

Given a circle C and an orthogonal arc $\overset{\frown}{RQ}$, then the two parts of the circle cut by the orthogonal arc are related by inversion in the arc: the right part of the circle below is the inverse of the left part, because of the property that circles orthogonal to the inversion circle remain invariant, apart from mirroring. In particular P is inverted into P^i on the other side of the circle circumference.



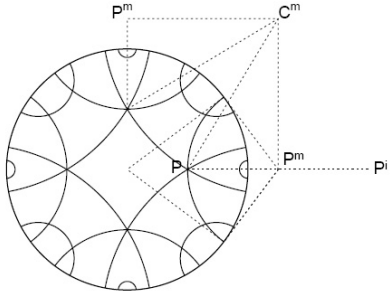
Escher's grids

Escher in his Circle Limits used grids orthogonal to the circle, except in his Circle Limit III, where the grid cuts the boundary at $\approx 80°$, as mentioned by Coxeter.

In fact finding a method to draw such grids was the incentive to this work on Circle Inversions.

It looked like that I needed the functionality to draw an orthogonal circle to a circle, which also passes through two prescribed points within the circle. It turned out that the picture can be parametrized by the radius of the circumscribing circle and *one* suitable chosen point within the circle, namely where the orthogonal arcs cross each other. The rest is implicit by the symmetry of the figure, which does not surprise me in Escher's drawings.

Let the radius of the circle be r and the parameter P, the inside point, be specified by P= (.6r, 0). Then the inverse of P is $P^i \approx (1.66r, 0)$ with mean $P^m = [P, P^i] \approx (1.13r, 0)$. The symmetrical orthogonal circle through P has centre P^m and radius ≈0.53r. Rotation over 90° yields its symmetrical counterpart. The orthogonal circle through P and P rotated over 90° has centre (1.13r, 1.13r), approximately.



In Circle Limit III the arcs intersect the boundary at ≈80°. Coxeter calls these lines ‘ . . . one of the branches of an equidistant curve.’ The hyperbolic and Escher-like grid depends on r and one data point P and is obtained by drawing arcs orthogonal to the circumference which pass through this point.

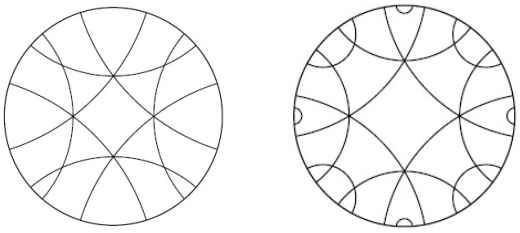
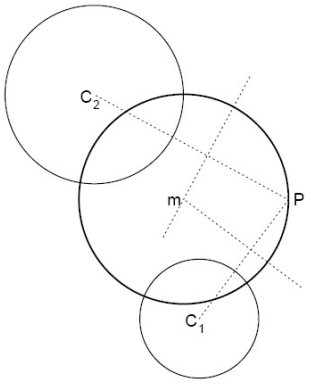


Figure 5. Hyperbolic Escher-like grid

In the right figure arcs have been inverted into opposite arcs, as second step towards a Circle Limit grid.

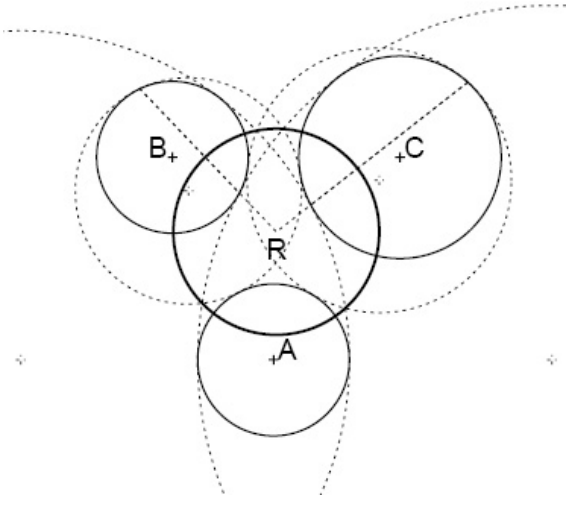
```
%!PS Orthogonal arcs through one point within a circle. CGL March 2010
%%BoundingBox: 0 0 620 790

(C:\\PSlib\\PSlib.eps) run %PS inversion library

200 300 translate
/r 100 def
0 0 r 0 360 arc
gsave  stroke grestore
clip %later drawing shows only within the circle with radius r
/Px r 2.7 div def
/Py 0 def
/P {Px Py} def
P 0 0 r pointinversion
   /Qy exch def /Qx exch def /Q {Qx Qy} def
P Q middleperpendicularvar
   /mQy exch def /mQx exch def
   /mPy exch def /mPx exch def
   /mQ {mQx mQy}def /mP {mPx mPy}def
mP mQ mean
   /y exch def /x exch def
/r1 Px x sub Py y sub size def

4{x y r1 0 360 arc stroke
    90 rotate}repeat

/r2 Px x sub Py x sub size def
4{x x r2 0 360 arc stroke
    90 rotate}repeat
showpage
```

Circle orthogonal to two (distinct) circles To make the problem unique add to the conditions that the circle must pass through a point P. Find the inverses of this point in both circles. The intersection point of the middle perpendiculars of P with its inverses P_1^i and P_2^i is the centre of the orthogonal circle.



Circle orthogonal to three (distinct) circles: the radical circle The centre of the radical circle is called the radical centre of the three circles. On this configuration we can apply inversion geometry, to find the radical centre. Once we have found this centre we can draw the radical circle.

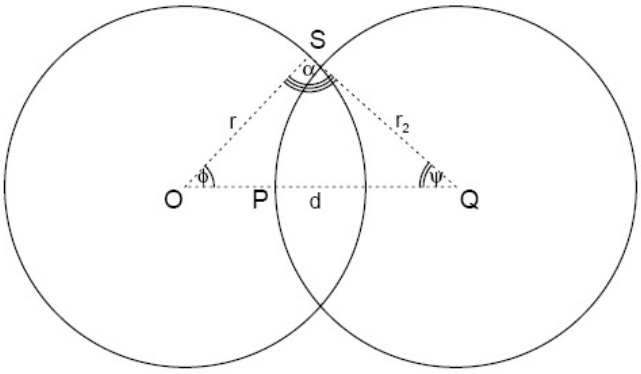
Inversion in the radical circle leaves the given circles unchanged,[27] but transforms the two conjugate solution circles into one another. Under the same inversion, the corresponding points of tangency of the two solution circles are transformed into one another.[28] Hence, the lines connecting these conjugate tangent points are invariant under the inversion; therefore, they must pass through the centre of inversion, which is the radical centre.[29]



Remark The in- and circumscribed circles as conjugate pair could equally-well have been used in the picture above.

Circle which intersects $C_{r,(0,0)}$ at an arbitrary angle and passes through a point P within $C_{r,(0,0)}$ The angle of intersection between curves is the angle between the tangents. For our case of intersecting circles we can use the equivalent of the angle between the radii to the intersection point.[30] This is handy, it keeps the explanatory pictures clear.

For intersection of orthogonal circles we had the circle inversion technique. In my trying, on the wrong tract as we we'll see later, to emulate Escher's Circle Limits III grid, I needed a general method for finding the circle, which cuts the given circle $C_{r,(0,0)}$ at 80°, and passes through P (on the x-axis) within the given circle.



Conditions for the $\angle\alpha$, i.e. $\angle$OSQ, as function of the radius r_2 of the 2nd circle, parametrized by r and P, read

$$\alpha_{r,\mathrm{P}}(r_2) = 180 - \phi - \psi$$
$$\phi = \arctan \frac{\mathrm{S}_y}{\mathrm{S}_x}$$
$$\psi = \arctan \frac{\mathrm{S}_y}{d - \mathrm{S}_x}$$

where the intersection point S of the circles is the solution with positive ordinate of the equations for the circles

$$x^2 + y^2 = r^2$$
$$(x-d)^2 + y^2 = r_2^2 \quad \text{with} \quad r_2 = d - p$$

two quadratic equations in two unknowns x and y. Complex.

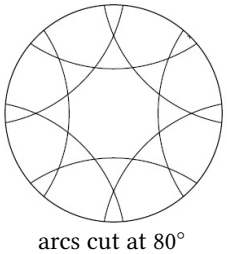
We can simplify, however, by subtracting equation 2 from 1, as done earlier in this note, and obtain

$$2d \cdot x = d^2 + r^2 - r_2^2 \quad \rightarrow \quad \mathrm{S}_x = \frac{d}{2} + \frac{r^2 - r_2^2}{2d}$$

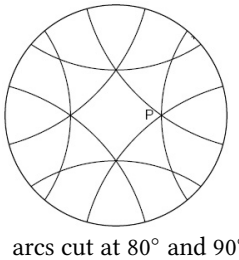
Substitution of this value in the equation for $C_{r,(0,0)}$, yields for the ordinate

$$\mathrm{S}_y = \sqrt{r^2 - \mathrm{S}_x^2}.$$

After finding iteratively the zero of the equation $\alpha_{r,\mathrm{P}}(r_2) - 80 = 0$, along similar lines as treated earlier in this note and implemented in the operator `circlesatalpha`, as given in Appendix I, we may draw the figures



arcs cut at 80°



arcs cut at 80° and 90°

For the left grid we rotated the arc through P over −45° 8 times. For the right grid we rotated P over −90° and drew an orthogonal arc through P and the rotated P. The latter arc was rotated 4 times over −90°. The right grid consists of arcs which cut the boundary at 90° and of arcs which the cut the boundary at 80°.

We have arrived at 3 grids
□ one where the arcs cut the boundary at 90°, see Figure 5
□ one where the arcs cut the boundary at 80°, left figure above, and
□ a grid where the arcs cut the boundary at 80° and at 90°, right figure above.

Which of these is the real Escher Circle Limit III grid?

Maybe the right grid above, where the arcs cut the boundary at 80° and at 90°. I prefer the earlier highly symmetric one, as displayed in Figure 5, where all the arcs cut the boundary orthogonally, and where the figure depends only on P (and r).

Did Escher miss something?

In the implementation the user is asked for an (estimate of the) upper bound of the radius of the wanted circle. As lower bound I assumed that the centre of the wanted circle is at the boundary of the main circle. This limits the use of `circlesatalpha`.

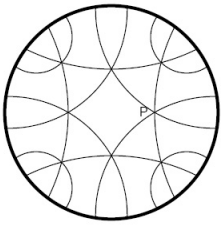
Remark1 The case with a circle $C_{r,(x,y)}$ and P arbitrarily within the circle, can make use of `circlesatalpha` by shifting the centre of the circle to the origin and rotating, the latter such that P will lie on the x-axis.

Remark2 I was on the wrong track, too much fixed at an angle of 80°. Coxeter's results of 1996 are by far superior, but ... the used math, especially `The Biquadratic field` $\mathbb{Q}(\sqrt{2}+\sqrt{3})$, and in general the used hyperbolic geometry, I'm not yet familiar with, alas. Nevertheless ... it illustrates the power of math.

Coxeter's solution

Coxeter, in *What Escher left unstated*, The Mathematical Intelligences, 18, 4, 1996, analysed Escher's Circle Limit III. He started from the rotational symmetry at P and derived (the parameters for) the grid.

Circle Limit III

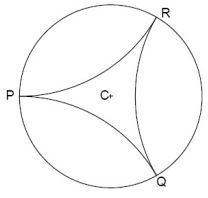


Coxeter's grid

Circle covered by triangles

Let us choose points P, Q and R equally distributed along the circumference of a circle. Draw the orthogonal arcs $\overset{\frown}{PQ}$, $\overset{\frown}{QR}$ and $\overset{\frown}{RP}$. A hyperbolic regular $\triangle$PQR is obtained, of which the sides are hyperbolic lines, often called d-lines of the so-called Poincaré disk in hyperbolic geometry.

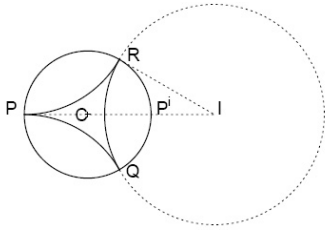
The triangle is also known under the name hypocycloid, which for the case of the regular $\triangle$ is obtained after rolling a circle with radius $\frac{1}{3}$R along the inside of the main circle with radius R. The triangle has sum 0 of the inner angles.



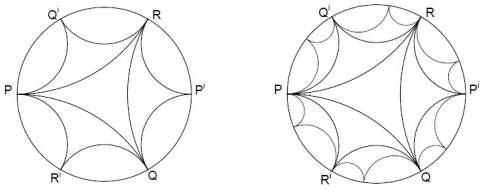
Interesting pictures are obtained when we invert the triangle repeatedly in its sides.

Inversion of P, Q and R in the opposing circular arcs yields the the points P^i, Q^i, R^i, which lie on the clipping circle C.

Proof Let us look at P, P^i and the midpoint of the inversion circle $I_{r\sqrt{3},(2r,0)}$, of which $\overset{\frown}{RQ}$ is an arc. Then $|\vec{IP}| \cdot |\vec{IP}^i| = r^2$, because the circles intersect orthogonally, i.e. $\vec{IR}$ is tangent to the clipping circle *C*.

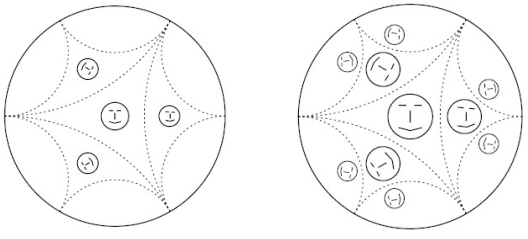


After inversion of P, Q and R, and drawing the (orthogonal) circular arcs, PQ^i, Q^iR et cetera, the clipping circle C is covered by four hyperbolically congruent triangles: PQR, and its three hyperbolically mirrored images PRQ^i, PQR^i, and RQP^i.



An interesting continuation is, see the above figure at the right, to invert P in R^iQ (and symmetrically Q in R^iP), Q in P^iR (and symmetrically R in P^iQ), R in Q^iP (and symmetrically P in Q^iR).[31] As result the clipping circle *C* is covered by ten hyperbolically congruent triangles.

Inversion of a smiley pattern can be done by use of the PostScript operator `pathforall`. This operator appends to the current path. It is not straightforward how to stroke the path created by `pathforall`, separately. I call the used technique[32] partial delayed execution, which can be useful as shown in this case. The pattern is inverted in two parts: eyes, nose and mouth by `pathforall` and the circumference by `circleinversion`.



In the right picture, with 2^{nd} level inversions, the operators `pointinversion` and `circleinversion` are invoke repeatedly: the invoke for the 1^{st} level inversion is immediately followed by the invoke for the 2^{nd} level inversion in the procedures for `pathforall`.

PS code with 1^{st} level inversions only. It demonstrates how to use fruitfully `pathforall`.

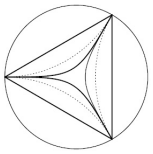
```
%!PS-Adobe-EPSF-3.0  pathforall use, cgl jan 2010
%%BoundingBox: 150 250 250 350

(C:\\PSlib\\PSlib.eps) run %library inversion operators, constants

200 300 translate
```

```
/rs 12.5 def % size smiley
r 0 moveto 0 0 r 0 360 arc % surrounding circle r=50
/smiley{% path of inner smiley
-5 5 moveto 3 0 rlineto
5 5 moveto -3 0 rlineto
0 2.5 moveto 0 -2.5 lineto
-5 -5 moveto 5 -1.25  rlineto 5 1.25 rlineto
}def
smiley % creates path of inner part of smiley
rs 0 moveto 0 0 rs 0 360 arc stroke % circumference of smiley
stroke % central smiley is drawn
%
/Ix 2r def /Iy 0 def /Ir rsqrt3 def % Inversion circle centre and radius
3{smiley  % creates path of inner smiley
[{ Ix Iy Ir pointinversion /moveto cvx}
 { Ix Iy Ir pointinversion /lineto cvx}
 { }
 { }
 pathforall
] cvx      % make array executable
newpath    % delete path of central smiley
exec       % path of inverted smiley is created (only)
stroke
 0 0 rs Ix Iy Ir circleinversion /ir exch def /iy exch def /ix exch def
newpath ix iy ir 0 360 arc stroke         % circumference of inverted smiley
Ix Iy 120 rot /Iy exch def /Ix exch def   % rotate centre of inversion circle
}repeat
%
/Mx r def /My rdsqrt3 def                 % centre of circles (sides triangle)
auxlin     % from libray: dashed auxiliary lines
3{newpath Ix Iy Ir 150 210 arc stroke     % arc of inversion circles
   120 rotate}repeat
/R {hr hr sqrt3 mul}def
6{R moveto Mx My My 150 270 arc stroke    % inverted sides
   60 rotate}repeat
showpage
```

If you, kind reader, can't resist the temptation to run the above (apparently complete) PostScript program, as was the case with Nico Temme, when I asked his opinion about this note, keep in mind that several predefined constants and operators from my `PSlib` library are used, some of which are discussed and given elsewhere in this note.

Continuation to the limit of the above processes will yield a clipping circle covered by smaller and smaller triangles, which form a grid, casu quo smaller and smaller smileys. The resulting pictures I call in the footsteps of M.C. Escher Circle Limits. Continuation to the limit is cumbersome, maybe less cumbersome than for Escher when he drew his artistic Circle Limits.

Inversion of hyperbolic arcs As pattern choose the lines which connect the corner points of the hypocycloid by arcs with their two control points (of the Bézier arc) at the centre of the circle. This looks like the Mercedes logo. Inversion of this logo in the dashed orthogonal circles yields straight lines, because the inverse of the centre of the circle is the mean of the corners of the hypocycloid. The `curve` procedure of `pathforall` is exercised.



I wanted to invert a 'fish' pattern in the hypocycloid (dashed) in the picture above. This result confirmed experimentally that Escher did not use circle inversion. He used hyperbolic rotations to 'copy' his patterns.

Circle through three points

A circle is usually defined by the set of points which lie at a constant distance from the circle centre, usually the origin.

$$\left\{z = (x, y) \;\middle|\; x^2 + y^2 = r^2\right\}$$

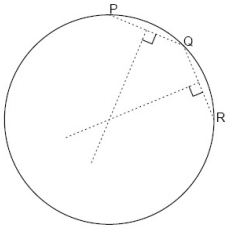
In the series http://www.pandd.demon.nl/complex1/hypm1.htm I came across another definition of the circle named after Apollonius: the set of points of which the quotient of the distance to two points is constant.

$$\left\{z = (x, y) \;\middle|\; \frac{|z - \mathrm{P}|}{|z - \mathrm{P}^i|} = \mathrm{c}\right\}$$

where P and P^i are inverse points towards the circle, and c a constant.

A circle is usually specified by the coordinates of its centre and its radius (three data), as is required for the PostScript operator `arc`.

Another description of a circle is by three points on the circumference.

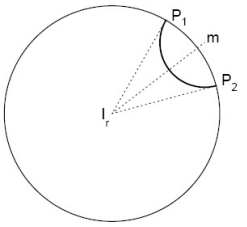


Below an operator is given for drawing a circle given three points on the circumference.[33]

```
/threepointscircle% P, Q, R on stack i.e. x1, y1, x2, y2, x3, y3
%==>
%x, y, r: centre and radius of circle
{0 begin
 /Ry exch def /Rx exch def /Qy exch def /Qx exch def /Py exch def /Px exch def
Px Py Qx Qy middleperpendicular
  /p1y exch def /p1x exch def /p2y exch def /p2x exch def
Qx Qy Rx Ry middleperpendicular
  /q1y exch def /q1x exch def /q2y exch def /q2x exch def
p1x p1y p2x p2y q1x q1y q2x q2y intersect /y exch def /x exch def
/r Px x sub Py y sub size def
x y r
end
}def
/threepointscircle load 0 15 dict put
```

The operators `middleperpendicular` and `intersect` will be discussed in Appendix I.

Orthogonal circle through 2 points on the inversion circle



The points P_1 and P_2 are specified by their polar coordinates: the radius r of the circle I_r through these points and their angles $\alpha 1$ respectively $\alpha 2$. The radius r_o and the centre (M_x, M_y) of the orthogonal circle are given by

$$\begin{aligned} r_o &= r \tan .5(\alpha_1 - \alpha_2) \\ M_x &= IM \cos .5(\alpha_1 + \alpha_2) \\ M_y &= IM \sin .5(\alpha_1 + \alpha_2) \\ IM &= \frac{r}{\cos .5(\alpha_1 - \alpha_2)}. \end{aligned}$$

The above is implemented in the operator `orthogonal` as given below, which is used in the section 'a circle covered by triangles.' Of the orthogonal circles only the clipped parts, which lie within the circle, are drawn. No explicit clipping. Assumed is that the circle centre, I_r, is at the origin.

```
/orthogonal % r phi1 phi2 on stack
%Purpose draw inner arc of orthogonal circle through
%(r cos(phi1), r sin(phi1)) and (r cos(phi2), r sin(phi2))
%assumed is that the circle has its centre at the origin.
{0 begin
/phi2 exch def  /phi1 exch def  /r exch def
/IM  r phi1 phi2 sub .5 mul cos div def     % auxiliary
/xP {r phi1 cos mul r phi1 sin mul}def      % shorthand
/mphi12 phi1 phi2 add .5 mul def            % mean of angles
/xm12 IM mphi12 cos mul def                 % x coord circle centre
/ym12 IM mphi12 sin  mul def                % y coord circle centre
/rm12 phi1 phi2 sub .5 mul dup sin exch cos div r mul def
                                            % r*tan.5(phi1-phi2) is  radius
xP moveto %move to centre of orthogonal circle
newpath xm12 ym12 rm12 phi1 90 add dup 180 phi1 sub phi2 add add arc stroke
end}
def
/orthogonal load 0 9 dict put
```

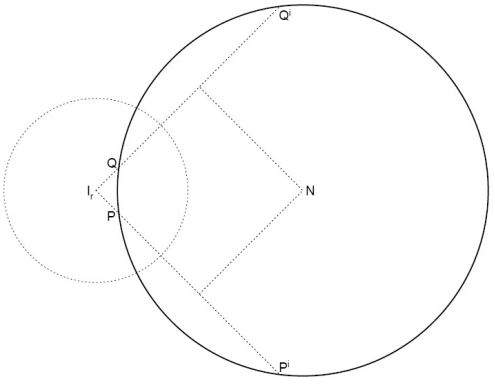
I needed the above for drawing orthogonal circles through two points on the circumference of a circle.

Remark It looked like that small grid arcs in Circle Limit III were drawn in this way. Not true. Coxeter proved that the centre of the small circle is off the circumference by a few per mille: at distance $.9981r$ from the centre of the main circle. Not visible, I guess, so my assumption was not too bad.

Orthogonal circle through 2 points within a circle

Invert both points P and Q in the circle I_r. The intersection point of the middle perpendiculars of $\vec{PP}^i$ and $\vec{QQ}^i$ is the centre of the required circle.

Proof IQ.IQ^i = r^2 by definition (of inversion). The equation also holds for two lines through I, of which one intersects the circle and the other is tangent to the circle. Therefore the line from I to the intersection points of the two circles is tangent to the other circle, and hence the circles are orthogonal.



The picture has been obtained by the following operator with P=.25r(1, -1) and Q=.25r(1, 1).

```
/twopointsincircle
%P1x P1y P2x P2y: points within the circle
%r         : radius of inversion circle (at centre)
{0 begin
/r exch def
/Py exch def /Px exch def /Qy exch def /Qx exch def
/P {Px Py}def /Q {Qx Qy}def
gsave auxlin
0 0  r 0 360 arc stroke
0 0 moveto -5 -5 rmoveto (I) show
                    0 -2 rmoveto gsave H5pt setfont (r) show grestore
P moveto -9 -7 rmoveto (P) show
Q moveto -10 -0 rmoveto (Q) show
P 0 0 r pointinversion /py exch def /px exch def
    px py moveto  gsave -1 1 rmoveto  H7pt setfont(P) show
                                      0 3 rmoveto H5pt setfont (i) show
                            grestore
Q 0 0 r  pointinversion /qy exch def /qx exch def
    qx qy moveto  gsave -1 -9 rmoveto H7pt setfont(Q) show
                                       0 3 rmoveto H5pt setfont (i) show
                            grestore
0 0  moveto px py lineto stroke
0 0 moveto qx qy lineto stroke
P px py middleperpendicular /pmy exch def /pmx exch def /py exch def /px exch def
Q qx qy middleperpendicular /qmy exch def /qmx exch def /qy exch def /qx exch def
pmx pmy px py qmx qmy qx qy intersect /y exch def /x exch def
pmx pmy moveto x y lineto  qmx qmy lineto stroke
/r Px x sub Py y sub size def
x y moveto 1 -2 rmoveto (N) H7pt setfont show
grestore
newpath x y r 0 360 arc stroke%orthogonal circle
end
}def
/twopointsincircle load 0 21 dict put
```

I thought that I needed this operator for drawing Escher's Circle Limit grids. Closer inspection of Escher's Circle Limit III grid yielded that only one point within the circle is sufficient for drawing the grid. The same holds for Circle Limit I, although I'm puzzled about which grid he used, and why.

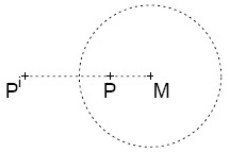
Circle prescribed by a point and its inverse towards a circle with radius r

No general operator is provided, just the principle is illustrated. Assume that one inversion point, P^i, is at the origin and the other, P, at (30, 0), and the radius r = 25. Let us suppose that the midpoint m of the circle is x away from P: M= $(30 + x, 0)$. The equation for x reads

$$x(x + \mathrm{P}) = \mathrm{r}^2 \qquad \text{with} \qquad \mathrm{P} = 30, \mathrm{r} = 25$$

The centre M follows from the solution x of the above quadratic equation

$$x = .5(\sqrt{(900 + 2500} - 30) \approx 14.15 \quad \rightarrow \quad \mathrm{M} \approx 44.15$$



The Apollonius constant $\mathrm{c} = \frac{|z-\mathrm{P}|}{|z-\mathrm{P}^i|}$ equals 39.15/69.15 ≈ .56 ≈ 10.85/19.15, where *en passant* we verified the property that the inverse points are the points of Apollonius.

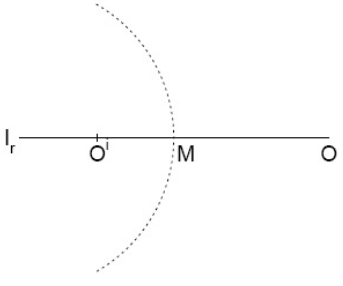
Reflection is not inversion

I asked myself the question whether a relation exist between circle inversion and reflection in a convex circular mirror, as arc of the inversion circle. In order to illustrate the difference I restricted myself to points on the real axis.

From the definition of inversion $\mathrm{IO} \cdot \mathrm{IO}^i = \mathrm{r}^2$, we derive

$$\frac{1}{\mathrm{MO}^i} - \frac{1}{\mathrm{MO}} = \frac{1}{\mathrm{r}}$$

where MO is the distance from O to the mirror M, i.e. $\mathrm{MO} = \mathrm{OI}_r - \mathrm{r}$, MO^i is the distance from the mirror to the inverted point O^i, i.e. $\mathrm{MO}^i = \mathrm{r} - \mathrm{IO}^i$, and r is the radius of the inversion circle *I*. This resembles the optical formula for mirrors we learned at high school: 'the mirror formula' $\frac{1}{v} + \frac{1}{b} = \frac{1}{f}$.



Acknowledgements

This note is a sidestep of familiarizing myself with hyperbolic geometry, towards understanding and making (variants of) Escher's Circle Limits. *En passant* PostScript (library) operators emerged.

This note is published in the TEX community, because it is an example of minimal plain TEX markup and because it exercises PostScript for creating pictures to be included in AnyTEX documents.

Thank you Jonathan Kew for providing the TEXworks environment for systems such as Vista, of which I became aware at the EuroTEX2009, and which I received on the TEX Live 2009 DVD.

Wim W. Wilhelm is kindly acknowledged for his remarks on this paper and for his mentioning of `Scite` as a versatile editor for MetaPost.[34] He also drew my attention to the Mathematica reader for viewing Mathematica notebooks.

Henk Jansen traced that the algebraic solution for the problem of Apollonius as developed in this paper, is mentioned in the superb survey http://en.wikipedia.org/wiki/Problem_of_Apollonius.

Thank you Nico Temme for your suggestions and help. Nico communicated that he uses Pascal to create pictures, exports them in `.eps` format, eventually post-processes them in Adobe Illustrator, for inclusion in his LATEX documents. Apparently, there is no need for him to use or program in PostScript. In the past he gave me a copy of Coxeter's *What Escher left unstated*, and recently he handed me the Pythagoras publication.

The TEX-world creates pictures in MetaPost, even Don Knuth ... however, one can make use of the PostScript library (see Appendix II) in MetaPost, conform to my philosophy to create libraries at the lowest level for (re)use at higher levels.

Thank you Taco Hoekwater for your suggestion to discern between the denotation of a point and a circle, and for your work to procrust this note into MAPS format. Most of all thank you for prompting how to overload operators in PostScript. Your suggestion to release `PSlib` on the WWW is well-taken. I'll announce the release at the BachoTeX2010, but for the moment I don't know where to release it: maybe on NTG's site, maybe on CTAN.

The MAPS proofreader is kindly acknowledged for the improvements on the use of English.

Thank you Bogusław Jackowski for stressing the importance of legibility and quality of the illustrations, and for your advice: do realize the consequences of B&W print, when colour nuances are lost.

Conclusions

Note what a little math can do towards mean-and-lean PostScript code. Especially, Apollonius problem has been reduced to solving one quadratic equation.

It is amazing how much math is available for free on the WWW via Wikipedia and personal sites, and ... not difficult to spot with the use of the right keywords, thanks to search engines.

For education in Math it seems that Cabri and animated Java are indispensable.

I'm still puzzled by why Escher did not use the highly symmetrical orthogonal grid, as given in Figure 5, for his Circle Limit III.

The assembling, creation, testing and disseminating of PostScript operators in PostScript program libraries is strongly advocated, because it eases the use of PostScript, and can be included in MetaPost.[35]

New, I think, is the operator `solve33` for solving 3 linear equations in 3 unknowns. The various operators for the inversion are new also, I presume.

The pearl of the paper is twofold:

- the rediscovery that Apollonius problem is solved by the solution of a quadratic equation, and
- the operator `Apollonius`, which reflects this discovery and can be used to obtain all 8 solutions of Apollonius problem. New probably, so is its cousin `Apollonius2`.

A beneficial spin-off of this work is the emerge of a PostScript library.

I would welcome an extension of TEXworks with a PostScript IDE, meaning edit PostScript in the left pane and view the `.pdf` in the right pane. This also holds for MetaPost.

MetaPost2, especially the multi-length arithmetic, would facilitate the use of MetaPost. A MetaPost library would ease the use of MetaPost too, to start with Hobby's `boxes` and `graphs`. Maybe the use of the emerging PostScript library in MetaPost will help as well.

With respect to TEX markup I had to kludge (just a tiny bit) for aligned harpoons in $\vec{P}$ and $\vec{P}^i$. For the markup of WWW links the catcode of the underscore and the %-sign had to be changed into 12. For colouring text with square-root signs and such, pdfTEX requires to include `\pdfliteral{1 0 0 0 k }` as well as `\pdfliteral{1 0 0 0 K }`.

The `jpgD` macro for the markup of a displayed `.jpg` picture obtained its cousin: the macro `pdfD`. In BLUe the code for verbatims was adapted in order to align the comments in the PostScript code. For the slides I had to reinstate magnification, which pdfTEX has switched off, and initialize some settings appropriate for slides, and write `\nxt`, next slide, the analogon of `\newpage`. No special slide package needed.

The extension of circle inversions into sphere inversions might be interesting, but ... to explore the matter more advanced tools are needed. Mathematica?

Isn't it amazing, that the incentive to this work was that I did not know how to draw a circle orthogonal to another circle and that it ended up with a PostScript library operator which draws an orthogonal circle to three other distinct circles.

Appendix I: PostScript library

PostScript operators

I gathered my PostScript operators under Vista in the file `PSlib.eps`. An invoke of the library can be done via inclusion of the following in your PostScript program

```
(C:\\PSlib\\PSlib.eps) run %Files are on  C disk in  directory PSlib
```

For the moment, `PSlib` consists[36] of constants, names (and CMYK settings) for colours,[37] some of Adobe's Bluebook operators, next to operators I developed myself.

I don't know how MetaPost definitions can be translated into PostScript operators.

PostScript programs, which for example test the PostScript operators, as e.g. given in Adobe's Blue Book, I store in

```
C:\\PSlib\\PSprg.eps
```

A snapshot of the contents of my PostScript library

To start with I have included constants like `sqrt2`, `sqrt3`, `sqrt5`, `pi`, ...

I borrowed from `pdfcolor.tex` the names and values of the CMYK colours, to enhance compatibility of colours in PostScript graphics and in pdfTEX, e.g.

```
/cmykBlue{1 1 0 0}def
/Blue{ cmykBlue setcmykcolor } def
use: Blue...
```

I also included some 'predefined' fonts.[38] For use of a predefined font, make the font *current* by the literal for the predefined font name, e.g. H12pt for Helvetica 12 points, followed by `setfont`.

From Adobe I took over the operators given in the Blue book.

Also included are fractals, such as binary tree, H-fractal, Pythagorean tree, snowflake, fern, Julia fractal, Koch fractal, Levy fractal, in short rewrites of Hans Lauwerier's Basic programs in PostScript, if not from Peitgen c.s.

Some emulated art like Linear I and Linear II from Naum Gabo, and Escher's impossible cubes, next to a few à la Mondrian are also candidate for the library. Some work still has to be done to make them available as operators.

In the following I'll discuss some operators developed by me.

Length, or better called size in PostScript A too simple operator? Not so.

One must circumvent intermediate (numerical) overflow, and... use the stack only because of the limited size of the `dict stack`.[39] Moreover, the name `length` is already in use as a so-called polymorphic operator—takes different kinds of arguments—also called an overloaded operator in ADA, for example. We can redefine length, but ... then we have lost the existing meanings. For the moment, I don't know how to extend a polymorphic operator in PostScript with more meanings, so I had to choose another name.

$$
\begin{aligned}
|(x,y)| &= \sqrt{x^2+y^2} \\
&= |y|\sqrt{1+(x/y)^2} \qquad \text{numerically better if} \quad |y| \geq |x| \\
&= |x|\sqrt{1+(y/x)^2} \qquad \text{numerically better if} \quad |x| \geq |y|
\end{aligned}
$$

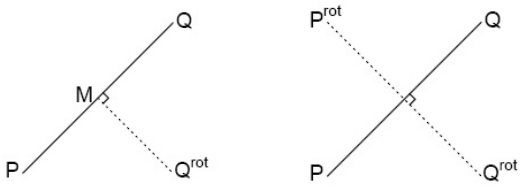
```
/size % x y ==> sqrt(x^2+y^2)
      % not robust against 0 0 on the stack
{abs dup  3 -1 roll abs dup 3 1 roll % |y| |x| |y| |x|
le {size}                           % |y|<=|x|
   {exch dup 3 1 roll               % |y| |x| |y|
     div                            % |y| |x|/|y|
     dup  mul 1 add sqrt mul
    }ifelse
}def
```

The operator is related to the polar coordinates (r, ϕ) of a point (x, y). In PostScript the angle ϕ can be obtained via the `atan` operator; for the size r one has to provide an operator oneself.

Overloading length While procrusting my contribution for MAPS, Taco Hoekwater came up with how to overload PostScript operators, which I incorporated in the PS library with result that the polymorphic `length` as well as `size` can be used.

```
%!PS Overloading length. Taco Hoekwater April 2010
/PSlength {length} bind def % save old meaning

/lengthdict 5 dict def
lengthdict /arraytype   {PSlength} put
lengthdict /dicttype    {PSlength} put
lengthdict /stringtype  {PSlength} put
lengthdict /integertype {size} put
lengthdict /realtype    {size} put

/length {
   lengthdict begin dup type exec end
} def

%Test
(whatever) length pstack pop
[1 2 3] length pstack pop
3 dict length pstack pop
3 4 length pstack pop
```

Middle perpendicular The problem is: given 2 points, say P and Q, draw the middle perpendicular.



When I was taught analytic geometry at high school, I don't think I would have come up with the nice mean-and-lean solution as implemented below in PostScript. There are two versions: one which yields PQ 90° rotated around the mean [P, Q], called `middleperpendicularvar`, and the other which yields the mean and one rotated endpoint, called `middleperpendicular`.

```
/middleperpendicularvar
% x1 y1 x2 y2 two points on stack
%==>
%the given points rotated 90 degrees around the mean.
{0 begin
/y2 exch def /x2 exch def /y1 exch def /x1 exch def
/xm x1 x2 add 2 div def /ym y1 y2 add 2 div def%middle
%translate (xm, ym) to Origin , rotate 90 degrees, translate back
/aux y1 ym sub neg xm add def
/y1 x1  xm sub     ym add def
aux y1
/aux y2 ym sub neg xm add def
/y2 x2  xm sub     ym add def
aux y2
end } def
/middleperpendicularvar load 0 10 dict put
```

Orthogonal marker The problem is to mark at the intersection point of two lines that the lines cross each other orthogonally.

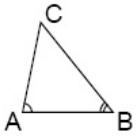
```
/ortho
%lx ly   point on left leg
%x y     cornerpoint
%s       size s of marker
%==>
%ortho symbol drawn of size s
{0 begin
 /s  exch def
 /y  exch def /x  exch def
 /ly exch def /lx exch def
 gsave x y translate
 lx x sub ly y sub atan neg  rotate
 0 s moveto s s lineto s 0 lineto stroke
 grestore
 end } def
/ortho load 0 5 dict put
```

See above in the diagram of the middle perpendicular how it looks. This code can also be used as a post-processing addition when you have converted MetaPost into PostScript.

Related to the orthogonal marker is the marking of angles by circular arcs.

Angle marker The problem is to mark an angle by circular arcs.

```
/anglemark
%l r cx  cy radius:  point left leg, point right leg, coordinates corner, radius
%==>
% angle marker in drawing
{0 begin  /r exch def
  /cy exch def
  /cx exch def
  /ry exch def
  /rx exch def
  /ly exch def
  /lx exch def
gsave
cx cy translate
newpath 0 0 r ry cy sub rx cx sub atan
                 ly cy sub lx cx sub atan arc stroke
grestore
end} def
/anglemark load 0 7 dict put
```



The preceding triangle is obtained by the following PostScript code

```
%!PS angle markers jan 2010, cgl
(C:\\PSlib\\PSlib.eps) run %PS library
50 50 translate
H14pt setfont
/A {0 0 } def
/B {50 0} def
/C {0 50} def
A moveto B lineto C lineto closepath stroke
A moveto  -10 -10   rmoveto (A) show
B moveto  2 -10     rmoveto (B) show
C moveto -10  0     rmoveto (C) show
A C B 5 anglemark
B A C 5 anglemark B A C 7 anglemark
C A 4 ortho
showpage
%EOF
```

The PostScript operator can also be used as a post-processing addition when you have converted MetaPost into PostScript. Note that in MetaPost the marking of an angle is not so straightforward, because MetaPost lacks the `arc` operator and the shifting of the device space functionality. Confer the above with the example in Hobby's report.

Mean of two points The problem is to calculate $.5[p_1, p_2]$.
Too trivial?

I found my code from more than a decade ago and considered it worthwhile to communicate, because the operator makes use of the stack only. Stack-oriented PostScript, different from the PostScript I used in my `Just a little bit of PostScript`, of old. No PostScript dictionary needed.

```
/mean
%p0 p1 on stack
%==>
%.5[p0, p1] i.e.\ x and y the coordinates of the mean
 {exch
  4 -1 roll add .5 mul
  3  1 roll add .5 mul}def
```

Neat, isn't it?

Rotation of a point Despite PostScript's functionality to rotate user space, I needed an operator to rotate just points. I chose conform the PostScript tradition that a positive angle rotates counterclockwise.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

```
/rot
%x y phi: point and angle of rotation (counterclockwise)
%==>
%x y coordinates of point after rotation
{ 0 begin /phi exch def /y exch def /x exch def
 /xaux x phi cos mul y phi sin mul sub def
 /y    x phi sin mul y phi cos mul add def
 /x xaux def
 x y
 end } def
```

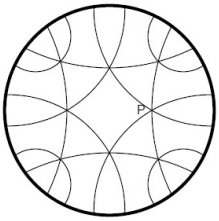
Binary tree The difference with the code in the EuroTEX2009 proceedings is the careful use of `currentpoint`, which resulted in a concise code.

```
/Bintree{% order -> balanced path of (2^order -1) leaves
         %Revised Jan 2010
E /order exch 1 sub def /y y 2 div def
order 1 ge {currentpoint
            N order Bintree
            moveto
            S  order Bintree}if
/order order 1 add def
/y y 2 mul def }def %end Bintree
```

H-fractal The operator is explained in the EuroTEX2009 proceedings.

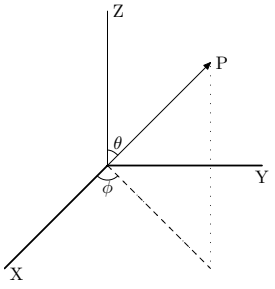
```
/Hfractal{/k exch def
   gsave draw
   /k k 2 mul 3 div def
   k 1 gt { 90 rotate k Hfractal
            -180 rotate k Hfractal}if
   /k k 3 mul 2 div def
   grestore}def
/draw{0 k rlineto
    currentpoint stroke translate
    0 0 moveto}def
```

Circle Limit III grid Coxeter, 1996, analysed Escher's Circle Limit III. He started from the rotational symmetry and derived (the parameters for) the grid.



Projection: ptp (point to pair) For projections I specify the graphics in 3D and project the data onto the plane by the operator `ptp` with viewing angles as parameters. The projection I also coded in MetaPost. The projection formula, with ϕ and θ viewing angles, reads

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -\cos\phi & \sin\phi & \\ -\sin\phi\sin\theta & -\cos\phi\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$




```
/ptp{% point x y z ==> x' y'
     % use: /pair { x y z ptp } def
     % parameters: a, b viewing angles
   0 begin
   /z exch def/y exch def/x exch def
   x neg a cos mul y a sin mul add
   x neg a sin mul b sin mul y neg a cos mul b sin mul add
   z b cos mul add
   end}def
/ptp load 0 3 dict put
```

Indispensable. A practical variant with **f**ixed viewing angles, `ptpf`, is part of the library too.

Intersection of line and circle A circle is given by a quadratic equation and a line by a linear equation. However, the intersection points of a line and a circle can be found elegantly after rotation. When there is no intersection a warning is supplied.

```
/lineintersectscircle%
% x1 y1 x2 y2:   points which specify the line
% mx my r:       centre and radius of circle
%==>
% x1, y1, x2 y2: coordinates of the intersection points if any.
{0 begin
/r exch def /my exch def /mx exch def
/y2 exch def /x2 exch def  /y1 exch def /x1 exch def
/angle y1 y2 sub x1 x2 sub atan 90 sub def
% shift upper point, i.e. centre of circle becomes 0 0
/y1 y1 my sub def /x1 x1 mx sub def
/lx  x1 angle cos mul y1 angle sin mul add def % abcis rotated point
```

```
lx abs r lt { % cuts the circle
  /yi r r mul lx dup mul sub sqrt def
  /x1 lx def /y1 yi def /x2 x1 def /y2 y1 neg def
  x1 y1 angle rot /y1 exch my add def /x1 exch mx add def
  x2 y2 angle rot /y2 exch my add def /x2 exch mx add def
  x1 y1 x2 y2
}{
  (No intersection point) print
}ifelse
end
} def
/lineintersectscircle load 0 20 dict put
```

Julia-fractal (due to Hans Lauwerier & Peitgen c.s.) A application of PostScript's `srand`, the random generator.

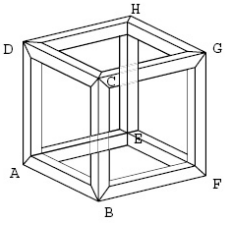
```
%!PS-Adobe- Julia sets, cgl May 97
%%Author: Peitgen e.a. (1992): Chaos and fractals. Springer-Verlag
%%BoundingBox: [-100 0 100 50]
300 300 translate
/Courier findfont 7 scalefont setfont
/s 50 def                       % scaling
/cr -1 def    /ci  0 def        %c as complex number
/cr  0 def    /ci  1 def        %c as complex number
/cr -.83 def  /ci  .16 def      %c as complex number
/x .25 def  /y 0 def /dofirst true def
/hrange 2 31 exp 1 sub .5 mul def
10 srand
8192{/a x cr sub def /b y ci sub def
a 0 gt{/x a a mul b b mul add sqrt a add .5 mul sqrt def
       /y b 2 x mul div def
      }
      {a 0 eq{b 0 eq{/x 0 def /y 0 def}
                    {/x b abs .5 mul sqrt def
                     /y b 2 x mul div def
                    }ifelse
             }
             {/y a a mul b b mul add sqrt a sub .5 mul sqrt def
              b 0 lt{/y y neg def}if
              /x b 2 y mul div def
             }ifelse
       }ifelse
%dofirst{/x  x .5 add def /dofirst false def}if
x s mul y s mul neg moveto (.) show
```

```
x s mul neg y s mul moveto (.) show
rand hrange gt{/x x neg def /y y neg def}if
}repeat
stroke
```

Uniform deviate delivers a pseudo-random number within the interval (0, argument).

Mondrian Operator for generating personalized Mondrian-esque birthday presents, with a Square, Oval or Lozenge cadre at choice.

Escher's impossible cube The code is too lengthy and boring to be included here.



Intersection of two lines I wrote these operators, `intersect`, `makecoeff`, `solveit` more than a decade ago. Only the stack is used. The lines are characterized by two points on each line, and these points have to be supplied on the stack for the invoke of `intersect`.

The points are transformed into an equation of the line

$$ax + by = e$$

by `makecoeff`. `solveit` solves the equations in the form delivered by `makecoeff`.

```
/makecoef
%z1 z2 -> e a b
{4 copy          %x1 y1 x2 y2 x1 y1 x2 y2
4 -1 roll mul    %x1 y1 x2 y2 y1 x2 (y2x1)
3 1 roll mul sub %x1 y1 x2 y2 (y2x1-y1x2)
5 1 roll 3 -1 roll sub
                 %(y2x1-y1x2) x1 x2 y2-y1
3 1 roll sub     %(y2x1-y1x2) y2-y1 x1-x2
}def
/solve22{%e a b f c d  -> x y,
         %intermediate p is pivot
%Equations: ax + by = e
%           cx + dy = f
%pivot handling  %e a b f c d
1 index abs      %e a b f c d |c|
5 index abs      %e a b f c d |c| |a|
gt {6 3 roll} if  %exchange 'equations'
%stack: e a b f c d or f c d e a b,
%first is in comments below
exch 4 index     %e a b f d c a
div              %e a b f d p
6 -1 roll dup 6 1 roll 3 1 roll
                 %a e b f e d p
4 index exch     %a e b f e d b p
dup 4 1 roll     %a e b f e p d b p
mul sub          %a e b f e p (d-b.p)
4 1 roll mul sub  exch div
```

```
%a e b (f-e.p)/(d-b.p) = a e b y
dup 5 1 roll mul sub exch div exch
%stack: x y
}def
/intersect {%p1 p2 p3 p4 -> x y
makecoef 7 3 roll
makecoef
solve22
}def
/mean{%p0 p1 on stack -> .5[p0, p1]
      exch 4 -1 roll add .5 mul
           3  1 roll add .5 mul}def
```

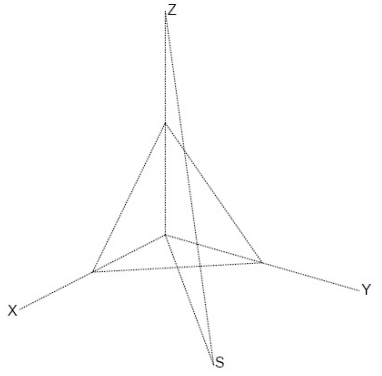
Intersection of planes `solve33` can be used to calculate the intersection point of the planes specified in matrix notation, for example as

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

The invoke

```
0 0  0 1
0 1 -1 0
1 1  1 1 solve33 /z exch def /y exch def /x exch def /determinant exch def
```

yields $(x, y, z) = (0.5, 0.5, 0)$ with determinant 2.[40]



The following operator `solve33` was used. In general `solve33` solves a linear 3x3-system, i.e. 3 equations.

```
/solve33{0 begin
%Purpose: Solve x y r from
% / x11 x12 x13 \   /x\        /rh1\
% | x21 x22 x23  | | y |  =   | rh2 |
% \ x31 x32 x33 /   \r/        \rh3/
%Input stack
%rh1 x11 x12 x13
%rh2 x21 x22 x23
%rh3 x31 x32 x33
%==>
%solution: determinant x y r
  /x33 exch def /x32 exch def /x31 exch def /rh3 exch def
  /x23 exch def /x22 exch def /x21 exch def /rh2 exch def
```

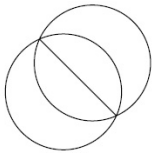
```
  /x13 exch def /x12 exch def /x11 exch def /rh1 exch def
  %calculation determinant
  /determinant x11 x22 x33 mul x23 x32 mul sub mul
               x12 x21 x33 mul x23 x31 mul sub mul sub
               x13 x21 x32 mul x22 x31 mul sub mul add def
%elimination last column, bottom up, to keep x,y as unknowns of
% 2X2-system
%make x33 biggest element, the pivot by exchanging rows
/max x33 abs def
  x13 abs max gt {/max x13 abs def} if
  x23 abs max gt {/max x23 abs def} if
  x13 abs max eq {%exchange row 1 and 3
  %10 -40 moveto (row 13 exchanged) show
          /aux x11 def /x11 x31 def /x31 aux def
         /aux x12 def /x12 x32 def /x32 aux def
         /aux x13 def /x13 x33 def /x33 aux def
         /aux rh1 def /rh1 rh3 def /rh3 aux def} if
  x23 abs max eq {%exchange row 2 and 3
  %10 -52 moveto (row 23 exchanged) show
          /aux x21 def /x21 x31 def /x31 aux def
         /aux x22 def /x22 x32 def /x32 aux def
         /aux x23 def /x23 x33 def /x33 aux def
         /aux rh2 def /rh2 rh3 def /rh3 aux def} if
%subtract row 3 times f from row 1
/f x13 x33 div def                          % x13/x33
/x11 x11 f x31 mul sub def                  % x11:=x11 - f*x31
/x12 x12 f x32 mul sub def                  % x12:=x12 - f*x32
/rh1 rh1 f rh3 mul sub def                  % rh1:=rh1 - f*rh3
%subtract row 3 times f from row 2
/f x23 x33 div def                          % x23/x33
/x21 x21 f x31 mul sub def                  % x21:=x21 - f*x31
/x22 x22 f x32 mul sub def                  % x22:=x22 - f*x32
/rh2 rh2 f rh3 mul sub def                  % rh2:=rh2 - f*rh3
%solve 2X2 subsystem
%gsave
%40 100 translate
%10   0 moveto (x11=) show x11 nstr cvs show
%              (   x12=) show x12 nstr cvs show
%             (      rh1=) show rh1 nstr cvs show
%10 -12 moveto (x21=) show x21  nstr cvs show
%              (   x22=) show x22  nstr cvs show
%             (     rh2=) show rh2 nstr cvs show
%10 -24 moveto (x31=) show x31  nstr cvs show
%              (   x32=) show x32  nstr cvs show
%             (   x33=) show x33  nstr cvs show
%             (   rh3=) show rh3 nstr cvs show
%grestore
%  / x11  x12 \   /x\       /rh1\
% |            | |   | =   |     |
%  \ x21  x22 /   \y/       \rh2/
rh1 x11 x12
rh2 x21 x22 solve22 /y exch def /x exch def
/rxy rh3
   x31 x mul x32 y mul add  sub
```

```
   x33 div  def
%
determinant x y rxy
end} def
/solve33 load 0 53 dict put
```

A handy and necessary feature is that the value of the determinant is delivered.

Intersection of two circles with equal radii The mean of the circle centres is translated to the origin. Rotate around the mean such that the line between the circle centres becomes the x-axis. The origin is the abscissa of the intersection point, calculate the ordinate and transform (rotate and translate) back.

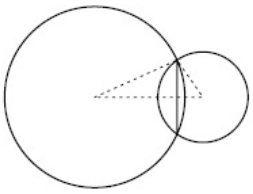
I needed this operator to determine the intersection of two grid lines in Escher's Circle Limit I.



```
/equalcirclesintersection % Purpose
                          % Intersection points of two circles with equal radii
% x1 y2 x2 y2 r:  centres of  circles and radius
%==>
% s1x s1y s2x s2y: two intersection points
{0 begin
/r exch def /y2 exch def /x2 exch def /y1 exch def /x1 exch def
x1 y1 x2 y2 mean /ym exch def /xm exch def
gsave
%translate mean to origin
/x1 x1 xm sub def /y1 y1 ym sub def
/x2 x2 xm sub def /y2 y2 ym sub def
%rotate such that line between centres coincides with the x-axis
/angle y2 x2 atan def
x1 y1  angle neg rot /y1 exch def /x1 exch def
/s1x 0 def /s1y r dup mul x1 dup mul sub sqrt def
/s2x 0 def /s2y s1y neg def
s1x s1y angle rot /s1y exch ym add def /s1x exch xm add def
s2x s2y angle rot /s2y exch ym add def /s2x exch xm add def
s1x s1y s2x s2y
end}def
/equalcirclesintersection load 0 22 dict put
```

Intersection of two circles The basic situation of two circles along the x-axis is solved. The sides of the triangle are known: r1, r2 and d, the distance. For the ordinate of the intersection point we can use from planimetry the formula for the perpendicular from the top on the basis (x-axis)

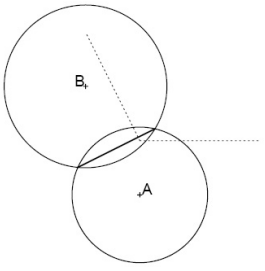
$$h = \frac{2}{d}\sqrt{s(s-d)(s-r1)(s-r2)} \qquad \text{with} \qquad s = \frac{r1+r2+d}{2}.$$



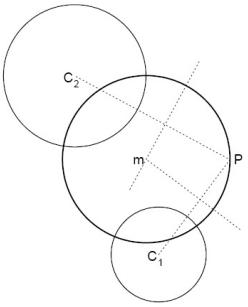
No need for solving quadratic equations. I thought that I needed this operator for finding the radical circle of three circles, not so because in that way it is an ill-posed problem. The operator maybe useful, nonetheless.

Intersection of two circles, general case The calculation is reduced to the special case by transformation and rotation. The situation of tangency has been addressed

$$|d - (r_1 + r_2)| <= \text{eps} \qquad \text{with} \qquad d = \|C_1 - C_2\| \quad \text{eps} = 0.00001.$$

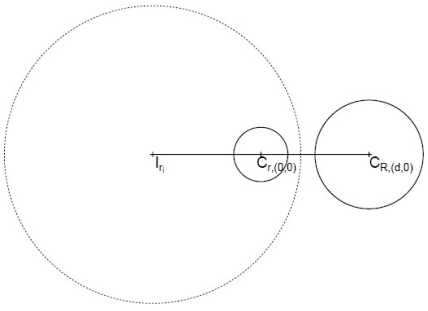


Circle orthogonal to two circles and passes through P The centre of the circle, m, is the intersection of the middle perpendiculars of P and the inverse points towards C_1 and C_2.

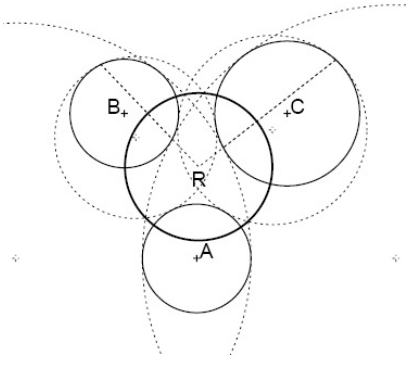


Inversion circle to conjugate circles, special case The problem is: given two distinct circles find the inversion circle which transforms the circles in each other.

```
/twoconjugatecircles2inversioncircle
% r: first circle  Cr(0, 0)
% d R: second cirlce CR(d, 0)
%==>
% x r: inversion circle Cr(x, 0)
{0 begin
/R exch def /d exch def /r exch def
/Ix d r mul R r sub div def
/ri Ix r add Ix d add R sub mul sqrt def %radius of inversion circle
Ix neg ri
end} def
/twoconjugatecircles2inversioncircle load 0 5 dict put
```



Circle orthogonal to three circles: radical circle A culmination of drawing orthogonal circles and the use of circle inversion. Quite a few operators are used such as `Apollonius`.

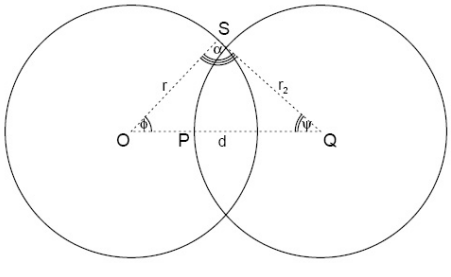


Interesting is that tangent point of two touching circles have to be determined as an intermediate step, which is an ill-posed problem if done by calculating the intersection of the two circles. I don't know of a situation where an ill-posed problem can so gracefully be circumvented. In this case it is better to determine the intersection of the line which connects the centre of a circle A, say, and the centre of the corresponding Apollonius circle with the circle A.

Not only is the determination of the intersection points of a line with a circle simpler than the determination of the intersection points of two circles, but in this case it is also much better conditioned. An educational pearl.

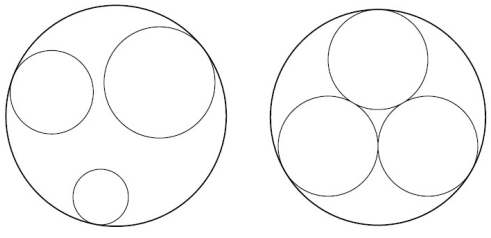
An ill-posed problem is not to be confused with an ill-conditioned solution technique. For example in solving a system of linear equations the complete or partial pivoting strategy is a much better conditioned numerical method that just Gaussian elimination, for the same problem.

Circle through P which intersects the given circle $C_{r,(0,0)}$ at the angle alpha The problem is to find the circle which intersects the circle $C_{r,(0,0)}$ at the given angle alpha and passes through a point within $C_{r,(0,0)}$.



```
/circlesatalpha %Purpose: construct circle which
                %cuts circle C{r(0,0)} at angle and passes through P
                %Looked for centre > r
%r px alpha rmax: radius main circle, coordinate P at x axis , angle, maximum r
%==>
% coordinate of centre along x-axis, radius, iteration cnt, angle:
%      mx  r cnt angle
{0 begin
/rmax exch def /alpha exch def
/px exch def /r exch def
%bounds for (abscis of) centre of circle
/mxr rmax def /mxl r def
%iteration prerequisites
/nmax 25   def % maximum number of iterations (safety)
/eps 0.001 def % required absolute precision
/cnt 0 def     % maintains number of iterations
%iteration
1 1 nmax{/cnt cnt 1 add def
/mx mxr mxl add 2 div def % bisection
/ri mx px sub def         % radius of circle through (px, 0), centre (mx,0)
%intersection point as function of ri
/r21 {ri r sub } def
/mr21 {ri r add 2 div } def
/xs {mx 2 div r21 mr21 mul mx div sub} def
/ys {r xs sub r xs add mul sqrt } def
/phi ys xs atan def
/psi ys mx xs sub atan def
/angle 180 phi sub psi sub def
angle alpha gt
{/mxr mx def}
{/mxl mx def} ifelse
mxr mxl sub abs eps lt {exit} if
}repeat
mx ri cnt angle
end}def
/circlesatalpha load 0 20 dict put
```

In- and circumscribed circle The problem is: given three distinct circles find the inscribed and circumscribed circle.

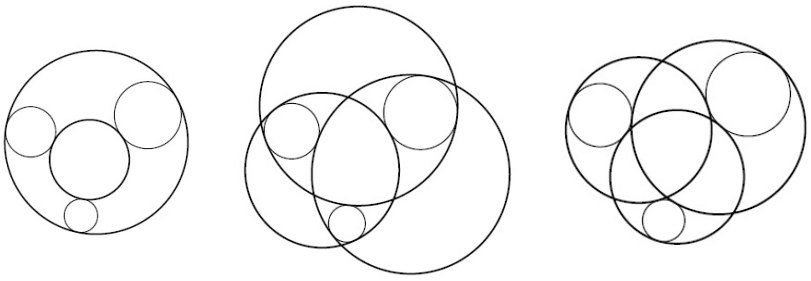


```
/circumscribed
%Purpose: Calculate circumscribed circle given three disjunct circles
% x1 y1 r1, x2 y2 r2, x3 y3 r3: centres and radii of three (disjunt) circles
%==>
% x y r:                        midpoint and radius of the inscribed circle
{0 begin
/r3 exch def /y3 exch def /x3 exch def
```

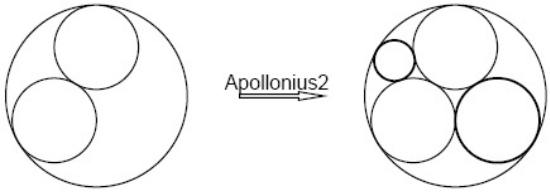
```
/r2 exch def /y2 exch def /x2 exch def
/r1 exch def /y1 exch def /x1 exch def
%auxiliary data
/x21 x2 x1 sub def /x31 x3 x1 sub def  /x32 x3 x2 sub def
/y21 y2 y1 sub def /y31 y3 y1 sub def  /y32 y3 y2 sub def
/r21 r2 r1 sub def /r31 r3 r1 sub def  /r32 r3 r2 sub def
/mx21 x2 x1 add 2 div def /mx31 x3 x1 add 2 div def /mx32 x3 x2 add 2 div def
/my21 y2 y1 add 2 div def /my31 y3 y1 add 2 div def /my32 y3 y2 add 2 div def
/mr21 r2 r1 add 2 div def /mr31 r3 r1 add 2 div def /mr32 r3 r2 add 2 div def
/g21 x21 mx21 mul y21 my21 mul add r21 mr21 mul sub def
/g32 x32 mx32 mul y32 my32 mul add r32 mr32 mul sub def

%exchange rows if |a21|> |a11| i.e. |x32| > |x21|
x32 abs x21 abs gt
{/aux x32 def /x32 x21 def /x21 aux def
 /aux y32 def /y32 y21 def /y21 aux def
 /aux r32 def /r32 r21 def /r21 aux def
 /aux g32 def /g32 g21 def /g21 aux def
 %gsave 0 30 moveto (rows exchanged) H12pt setfont show grestore
 } if

%express equations for x and  y as function of r
/p  x32 x21 div def
/a22 y32 p y21 mul sub def
/a2 r32 p r21 neg mul  add a22 div def
/b2 g32 p g21 mul sub a22 div def
/a1 a2 y21 mul r21 sub neg x21 div def
/b1 g21 y21 b2 mul sub x21 div def

/x {a1 r mul b1 add} def
/y {a2 r mul b2 add} def

%coefficients of quadratic equatio A*r^2 -2B*r + C = 0
/A a1 dup mul a2 dup mul add 1 sub def
/B r3 neg a1 b1 x3 sub mul sub
       a2 b2 y3 sub mul sub  def
/C b1 x3 sub dup mul b2 y3 sub dup mul add r3 neg dup mul sub def
/eps 0.000001 def
A abs eps lt
{/r C B div 2 div def
 gsave 0 30 moveto (A < .000001) show grestore}%warning
{/r B A div dup dup mul C A div sub sqrt add def}
 ifelse
x y r
}def
/circumscribed load 0 65 dict put
```

The operator `inscribed` is highly similar, only opposite signs for the $r_k, k = 1, 2, 3$ have to be accounted for.

The operator `Apollonius` is a general, unifying alias for the operator `inscribed` or `circumscribed`. The operator can be used to give each of the `8` solutions of the Apollonius circles for three distinct circles.



The operator `Apollonius2` is suited for a circle, with inside two other circles, the non-distinct case, and one wants the circles which touch the big circle from the inside and the smaller, inside circles from the outside.



With `Apollonius` and `Apollonius2` the beautiful pictures I borrowed from the WWW can be reproduced.

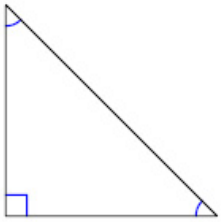
Appendix II: use of the PostScript library in MetaPost

MetaPost provides `special`⟨*string expression*⟩ for inclusion of PostScript.

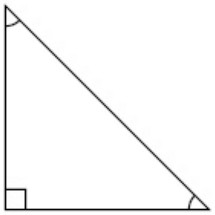
MetaPost with included PostScript and library use

```
if scantokens(mpversion) > 1.005:
  outputtemplate :=
else:
  filenametemplate
fi
"%j.eps";
special "(C:\\PSlib\\PSlib.eps) run ";
special "250 250 translate blue";
special "20 10 10 10 10 20 9 anglemark 10 10 10 20 20 10 9 anglemark ";
special "10 20 10 10 9 ortho ";
beginfig(0)
draw (10, 10)--(20, 10)--(10, 20)--cycle;
endfig;
end
```

The left figure below is obtained by the MetaPost program with use of PostScript as given above and the right figure is obtained by PostScript straight away, of which the code is given below. The difference in the pictures is the use of the colour blue (picture left) in the MetaPost program for the angle markers.



PostScript included in MetaPost



PostScript straightaway

Encouraging, for the use of the PostScript library operators in MetaPost.

Straightaway PostScript, with the use of symbolic names, for the example given above reads

```
%!PS angle markers March 2010, cgl
%%BoundingBox: 0 0 620 790
(C:\\PSlib\\PSlib.eps) run
250 250 translate
/r 100 def /2r 2 r mul def
/A {  r  r} def /B {2r  r} def /C { r 2r} def
A moveto B lineto C lineto closepath stroke
C A 10 ortho B A C 10 anglemark A C B 10 anglemark
showpage
```

Remark: I can't make use in MetaPost of symbolic names introduced by PostScript, though they are introduced by the inserted PostScript at the beginning of the delivered PostScript program. It looks like MetaPost does not take notice of the included PostScript and just passes it on. A pity.

My case rests, have fun and all the best.

Notes

1. I usually compose illustrations separately, fine-tune them, and then include them in a plain TEX document.
2. For direct use in LATEX there is the package `PSTricks` where PostScript graphics is encapsulated in LATEX commands. `PStool` provides a command to include `.eps` in pdfLaTEX. It facilitates to interface with a generic PostScript workflow, used to great effect in `PSTricks` and `psfrag`. This note is about creating pictures in PostScript and including them in the document after transforming to a format suitable for pdfTEX. Direct inclusion of a `.eps` picture is possible via `DVIPS` in the workflow `.tex` → `.dvi` → `.ps` → `.pdf`.
3. See http://www.acumentraining.com/acumenjournal.html for advanced worked out examples of PS and PDF programming.
4. To avoid the 'notch' `2 setlinecap` is used.
5. Some more work has to be done to make the operators given in this note robust.
6. I don't know how to achieve selective loading of library parts, so it seems that the library should be split up into parts in order that the appropriate part can be loaded. But ...maybe selective loading is no longer relevant with the huge size of today's internal memory.
7. Courtesy http://en.wikipedia.org/wiki/Inversive_geometry
8. For example use the formula for the perpendicular $h_d = \frac{r}{d}\sqrt{s(s-r)}, s = \frac{2d+r}{2}$, for the ordinate of S.
9. T is the point of the inversion circle where the tangent from P to the inversion circle touches the circle.
10. := denotes assignation.
11. After rotation the endpoint of the perpendicular is the abscissa of the rotated P_1 (or P_2).
12. See http://en.wikipedia.org/wiki/Peaucellier-Lipkin_linkage for the proof and an animation of the movements.
13. Note: the inversion of the centre of the circle to be inverted is not the centre of the inverted circle.
14. See my Tiling in PostScript and MetaFont—Escher's wink. MAPS97.2. Explicit solutions exist.
15. Н К Рерих: Russian painter (1874-1947).
16. Pythagoras: Dutch Math journal for young people, especially high school students.
17. See for example *Courant&Robbins p161*. For other approaches consult the Wikipedia with keywords Apollonius problem.
18. Note that the centres are not altered and that the unknown circle is not used. We only have to keep track of what happens to the unknown circle during the transformations. Perhaps it is easier to start from the problem when two circles touch and ask oneself: what is the relation between the two problems? In terms of *Polya's How to solve it*: simplify the problem, and once solved, go from there to the original problem.
19. Note where U^i touches A^i.
20. Sharper bounds can be obtained via Descartes circle theorem.
21. A singular system because the third row of the matrix is the difference of the second row and the first row.
22. The Newton approach is by far superior, but in view of the simplest approach, see later, I did not pursue this further.
23. Note that in squaring we include the solution with distance $r_k - r$, meaning C_r touches C^k from the inside.
24. Unnecessary as we'll see later.
25. Note the negative radii.
26. Borrowed from http://en.wikipedia.org/wiki/Problem_of_Apollonius.
27. Orthogonal property.
28. As can be seen in the picture below, where conjugate circles cross at the boundary of the radical circle.
29. The determination of the tangent points is an ill-posed problem, i.e. numerical ill-conditioned, if done by finding the intersection of the Apollonius circle with the original circle

B, say. Better is to intersect the Apollonius circle by the line which connects the centre of the Apollonius circle and centre of the original circle B, say.
30. Rotate the tangents over 90 degrees and you have the radii.
31. The symmetrical arcs, Q in R^iP et cetera, have been omitted.
32. See http://www.acumentraining.com/acumenjournal.html.
33. The PostScript programming of this operator is not that trivial. We have to determine the intersection point of two lines, the middle perpendiculars of the chords, by solving 2 linear equations in 2 unknowns, in PostScript. This is susceptible to ill-conditioning when the points are close.
34. On the TEX Live DVD 2009, I found `MPedit` for editing MetaPost sources, but I could not view the resulting PostScript elegantly, not better than with Scite, so I stay with Scite.
35. I'm soliciting for help for extending the library, for making the library robust, and for thorough testing.
36. I'm looking for volunteers to help me in extending and maintaining the PostScript library.
37. Is there a standard for mnemotechnic names for colours and their CMYK values? Is the list given in the pdfTEX manual, of which the source is available in `pdfcolor.tex`, generally accepted, c.q. a de facto standard?
38. When the library is invoked the fonts are looked up in the `FontDirectory`, scaled to the given point size, and stored in the user directory associated with the names specified in the library.
39. The maximum size of the `dict stack` is 20.
40. Note that the rhs is given as first column, consistent with my already existing `solve22`.

April 2010
Kees van der Laan
Hunzeweg 57, 9893PB, Garnwerd, NL
kisa1@xs4all.nl

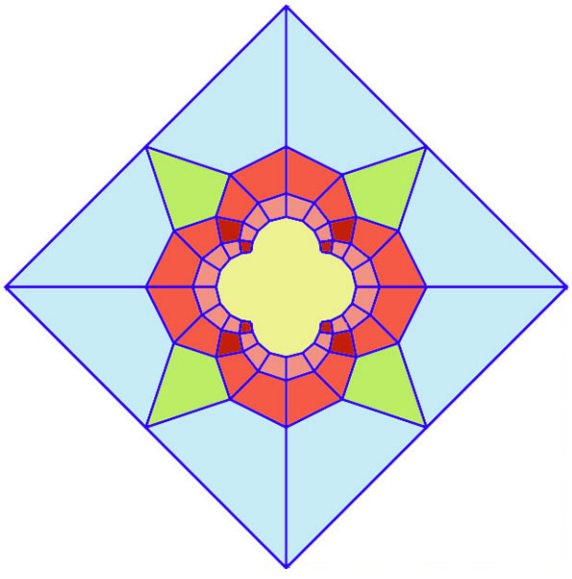


Figure 6. The stained glass window design.