MAPS

NUMMER 40 • VOORJAAR 2010

NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

Voorzitter
Taco Hoekwater
ntg-president@ntg.nl
Secretaris
Willi Egger
ntg-secretary@ntg.nl
Penningmeester
Ferdy Hanssen
ntg-treasurer@ntg.nl
Bestuurslid
Karel Wesseling
k.h.wesseling@planet.nl
Postadres
Nederlandstalige TEX Gebruikersgroep
Maasstraat 2
5836 BB Sambeek
Postgiro
1306238
t.n.v. NTG, Deil
BIC-code: PSTBNL21
IBAN-code: NL05PSTB0001306238
E-mail bestuur
ntg@ntg.nl
E-mail MAPS redactie
maps@ntg.nl
WWW
www.ntg.nl

De **Nederlandstalige TEX Gebruikersgroep (NTG)** is een vereniging die tot doel heeft de kennis en het gebruik van TEX te bevorderen. De NTG fungeert als een forum voor nieuwe ontwikkelingen met betrekking tot computergebaseerde document-opmaak in het algemeen en de ontwikkeling van 'TEX and friends' in het bijzonder. De doelstellingen probeert de NTG te realiseren door onder meer het uitwisselen van informatie, het organiseren van conferenties en symposia met betrekking tot TEX en daarmee verwante programmatuur.
De NTG biedt haar leden ondermeer:

- ☐ Tweemaal per jaar een NTG-bijeenkomst.
- ☐ Het NTG-tijdschrift MAPS.
- ☐ De 'TEX Live'-distributie op DVD/CDROM inclusief de complete CTAN software-archieven.
- ☐ Verschillende discussielijsten (mailing lists) over TEX-gerelateerde onderwerpen, zowel voor beginners als gevorderden, algemeen en specialistisch.
- ☐ De FTP server `ftp.ntg.nl` waarop vele honderden megabytes aan algemeen te gebruiken 'TEX-producten' staan.
- ☐ De WWW server `www.ntg.nl` waarop algemene informatie staat over de NTG, bijeenkomsten, publicaties en links naar andere TEX sites.
- ☐ Korting op (buitenlandse) TEX-conferenties en -cursussen en op het lidmaatschap van andere TEX-gebruikersgroepen.

Lid worden kan door overmaking van de verschuldigde contributie naar de NTG-giro (zie links); vermeld IBAN- zowel als SWIFT/BIC-code en selecteer shared cost. Daarnaast dient via `www.ntg.nl` een informatieformulier te worden ingevuld. Zonodig kan ook een papieren formulier bij het secretariaat worden opgevraagd.
De contributie bedraagt € 40; voor studenten geldt een tarief van € 20. Dit geeft alle lidmaatschapsvoordelen maar *geen stemrecht*. Een bewijs van inschrijving is vereist. Een gecombineerd NTG/TUG-lidmaatschap levert een korting van 10% op beide contributies op. De prijs in euro's wordt bepaald door de dollarkoers aan het begin van het jaar. De ongekorte TUG-contributie is momenteel $65.

MAPS bijdragen kunt u opsturen naar `maps@ntg.nl`, bij voorkeur in LATEX- of ConTEXt formaat. Bijdragen op alle niveaus van expertise zijn welkom.

Productie. De Maps wordt gezet met behulp van een LATEX class file en een ConTEXt module. Het pdf bestand voor de drukker wordt aangemaakt met behulp van pdftex 1.40.9 draaiend onder Linux 2.6. De gebruikte fonts zijn Linux Libertine, het niet-proportionele font Inconsolata, schreefloze fonts uit de Latin Modern collectie, en de Euler wiskunde fonts, alle vrij beschikbaar.

TEX is een door professor Donald E. Knuth ontwikkelde 'opmaaktaal' voor het letterzetten van documenten, een documentopmaaksysteem. Met TEX is het mogelijk om kwalitatief hoogstaand drukwerk te vervaardigen. Het is eveneens zeer geschikt voor formules in mathematische teksten.
Er is een aantal op TEX gebaseerde producten, waarmee ook de logische structuur van een document beschreven kan worden, met behoud van de letterzet-mogelijkheden van TEX. Voorbeelden zijn LATEX van Leslie Lamport, AMS-TEX van Michael Spivak, en ConTEXt van Hans Hagen.

Inhoudsopgave

Redactioneel

Het afgelopen jaar zijn beide servers die door de redactie worden gebruikt vervangen: eerst de interne server van de redactie, waarvan niet alleen de hardware vervangen is, maar die nu ook op een andere plaats hangt waar de internet verbinding stukken beter is. Later is ook de algemene ntg server vervangen, omdat de gratis web hosting bij de Universiteit Utrecht helaas ten einde kwam.

Zo'n verhuizing heeft heel wat voeten in aarde. Daarbij was er ook nog veel werk aan het afronden van de proceedings van EuroTEX 2009. Het gebrek aan mankracht bij de vereniging liet zich dan ook merken: de ntg website liep tot vorige maand aardig achter bij de werkelijkheid. Gelukkig is het stof nu weer gezakt, en er is een inhaalslag gemaakt om alles weer up-to-date te brengen.

Zo is, waarschijnlijk voor het eerst in de geschiedenis, deze Maps eerder on-line gepubliceerd dan op papier. Dankzij de inspanningen van Wybo Dekker staat de inhoudsopgave met de abstracts al sinds 26 April gereed, en de definitieve pdf bestanden gingen (*gaan*, terwijl ik dit schrijf) gelijktijdig naar de drukker én het alleen-voor-leden gedeelte van de ntg website (http://www.ntg.nl/voorleden.html).

En voor u ligt dan dus Maps 40. Deze Maps begint met een artikel van Hans Hagen. Een artikel met de naam 'The font name mess' wekt nu niet meteen een innovatieve indruk want laten we wel zijn, fonts zijn al een probleem sinds de allereerste keer dat een gebruiker probeerde een eigen font te gebruiken in TEX. En wat blijkt: nu de nieuwe generatie van TEX afgeleide engines in staat is om OpenType fonts te gebruiken, is het probleem niet zozeer opgelost, maar eerder verschoven en mogelijk nog ingewikkelder geworden. Hans laat zien hoe het probleem is opgelost voor LuaTEX en ConTEXt MkIV, maar in dit artikel is er ook ruim aandacht voor algemenere beschouwingen.

Wie afgelopen zomer niet naar de EuroTEX-conferentie geweest is heeft het wellicht gemist, maar Kees van der Laan is weer helemaal terug in de TEX-wereld. Niet alleen met een artikel in deze Maps, er ligt al een ander artikel klaar voor de volgende Maps. 'Circle Inversions' gaat wel wat meer over PostScript dan over TEX, maar dat maakt het zeker niet minder interessant. Kees zelf schrijft in zijn dankbetuiging (vrije vertaling door mij): *Deze notitie is een zijeffect van het mezelf vertrouwd maken met hyperbolische meetkunde, en dit weer om Escher's cirkellimieten te kunnen begrijpen en imiteren.* Dat is typerend voor Kees, geloof ik. Hij wil iets uitzoeken, moet er iets voor bijleren, van het een komt het ander, en het resultaat is een artikel van liefst 57 pagina's!

Het derde artikel, 'Grouping in hybrid environments', is weer van Hans. De titel van dit artikel doet misschien wat tekort aan de inhoud. Het gaat namelijk niet alleen over groepering van opties binnen LuaTEX, maar de daarbij gebruikte voorbeelden zijn een artikel op zich. Het onderlijnen van tekst is altijd een twijfelachtig punt geweest omdat het werd gezien als een overblijfsel van de typemachine periode. Maar als je 'onderlijnen' ook kunt doen met behoud van afbrekingen of met een achtergrondkleur in plaats van een streep, dan is het eindresultaat wel heel anders, en moet die mening misschien herzien worden.

'OpenType PostScript fonts with unusual units-per-em values' van Luigi Scarso geeft een goede kijk achter de schermen van LuaTEX. Niet alleen krijgt u daardoor een kijkje in de implementatie van een modern probleem, maar voor wie bekend is met de broncode van TEX zelf laat het ook goed de verschillen tussen TEX82 en LuaTEX zien.

Wij van de redaktie zijn heel blij dat de 'Nieuws van CTAN' rubriek van Piet van Oostrum terug is van weggeweest. Dit keer met veel aandacht voor pstricks, maar natuurlijk zijn er ook voor andere pakkketten bijgekomen of bijgewerkt.

Dan rest mij nog om u veel leesplezier te wensen, maar voor u deze bladzijde omslaat wil ik nog vragen of u eens na wilt denken over het zelf schrijven van een artikeltje voor de volgende Maps. De redactie kan altijd nieuwe artikelen gebruiken, ongeacht het verwachte niveau van de lezers. De volgende deadline is 1 oktober aanstaande.

Veel leesplezier!

Taco Hoekwater

The font name mess

Keywords
ConTeXt mkiv, luatex, font names

Introduction

When TEX came around it shipped with its own fonts. At that moment the TEX font universe was a small and well known territory. The 'only' hassle was that one needed to make sure that the right kind of bitmap was available for the printer.

When other languages than English came into the picture things became more complex as now fonts instances in specific encodings showed up. After a couple of years the by then standardised TEX distributions carried tens of thousands of font files. The reason for this was simple: TEX fonts could only have 256 characters and therefore there were quite some encodings. Also, large cjk fonts could easily have hundreds of metric files per font. Distributions also provide metrics for commercial fonts although I could never use them and as a result have many extra metric files in my personal trees (generated by TEXfont). (Distributions like TEXLive have between 50.000 and 100.000 files, but derivatives like the ConTEXt minimals are much smaller.)

At the input side many problems related to encodings were solved by Unicode. So, when the more Unicode aware fonts showed up, it looked like things would become easier. For instance, no longer were choices for encodings needed. Instead one had to choose features and enable languages and scripts and so the problem of the multitude of files was replaced by the necessity to know what some font actually provides. But still, for the average user it can be seen as an improvement.

A rather persistent problem remained, especially for those who want to use different fonts and or need to install fonts on the system that come from elsewhere (either free or commercial): the names used for fonts. You may argue that modern TEX engines and macro packages can make things easier, especially as one can call up fonts by their names instead of their file names, but actually the problem has worsened. With traditional TEX you definitely get an error when you mistype a file name or call for a font that is not on your system. The more modern TEX's macro packages can provide fall-back mechanisms and you can end up with something you didn't ask for.

For years one of the good things of TEX was its stability. If we forget about changes in content, macro packages and/or hyphenation patterns, documents could render more or less the same for years. This is because fonts didn't change. However, now that fonts are more complex, bugs gets fixed and thereby results can differ. Or, if you use platform fonts, your updated operating system might have new or even different variants. Or, if you access your fonts by font name, a lookup can resolve differently.

The main reason for this is that font names as well as file names of fonts are highly inconsistent across vendors, within vendors and platforms. As we have to deal with this matter, in MkIV we have several ways to address a font: by file name, by font name, and by specification. In the next sections I will describe all three.

Method 1: file

The most robust way to specify what fonts is to be used is the file name. This is done as follows:

```
\definefont[SomeFont][file:lmmono10-regular]
```

A file name lookup is case insensitive and the name you pass is exact. Of course the `file:` prefix (as with any prefix) can be used in font synonyms as well. You may add a suffix, so this is also valid:

```
\definefont[SomeFont][file:lmmono10-regular.otf]
```

By default ConTEXt will first look for an OpenType font so in both cases you will get such a font. But how do you know what the file name is? You can for instance check it out with:

```
mtxrun --script font --list --method=file --pattern="lm*mono" --all
```

This reports some information about the file, like the weight, style, width, font name, file name and optionally the subfont id and a mismatch between the analysed weight and the one mentioned by the font.

```
latinmodernmonolight      light   normal  normal     lmmonolt10regular          lmmonolt10-regular.otf
latinmodernmonoproplight  light   italic  normal     lmmonoproplt10oblique      lmmonoproplt10-oblique.otf
latinmodernmono           normal  normal  normal     lmmono9regular             lmmono9-regular.otf
latinmodernmonoprop       normal  italic  normal     lmmonoprop10oblique        lmmonoprop10-oblique.otf
latinmodernmono           normal  italic  normal     lmmono10italic             lmmono10-italic.otf
latinmodernmono           normal  normal  normal     lmmono8regular             lmmono8-regular.otf
latinmodernmonolightcond  light   italic  condensed  lmmonoltcond10oblique      lmmonoltcond10-oblique.otf
latinmodernmonolight      light   italic  normal     lmmonolt10oblique          lmmonolt10-oblique.otf
latinmodernmonolightcond  light   normal  condensed  lmmonoltcond10regular      lmmonoltcond10-regular.otf
latinmodernmonolight      bold    italic  normal     lmmonolt10boldoblique      lmmonolt10-boldoblique.otf
latinmodernmonocaps       normal  italic  normal     lmmonocaps10oblique        lmmonocaps10-oblique.otf
latinmodernmonoproplight  bold    italic  normal     lmmonoproplt10boldoblique  lmmonoproplt10-boldoblique.otf
latinmodernmonolight      bold    normal  normal     lmmonolt10bold             lmmonolt10-bold.otf
latinmodernmonoproplight  bold    normal  normal     lmmonoproplt10bold         lmmonoproplt10-bold.otf
latinmodernmonoslanted    normal  normal  normal     lmmonoslant10regular       lmmonoslant10-regular.otf
latinmodernmono           normal  normal  normal     lmmono12regular            lmmono12-regular.otf
latinmodernmonocaps       normal  normal  normal     lmmonocaps10regular        lmmonocaps10-regular.otf
latinmodernmonoprop       normal  normal  normal     lmmonoprop10regular        lmmonoprop10-regular.otf
latinmodernmono           normal  normal  normal     lmmono10regular            lmmono10-regular.otf
latinmodernmonoproplight  light   normal  normal     lmmonoproplt10regular      lmmonoproplt10-regular.otf
```

Method 2: name

Instead of lookup by file, you can also use names. In the font database we store references to the font name and full name as well as some composed names from information that comes with the font. This permits rather liberal naming and the main reason is that we can more easily look up fonts. In practice you will use names that are as close to the file name as possible.

```
mtxrun --script font --list --method=name --pattern="lmmono*regular" --all
```

This gives on my machine:

```
lmmono10regular         lmmono10regular         lmmono10-regular.otf
lmmono12regular         lmmono12regular         lmmono12-regular.otf
lmmono8regular          lmmono8regular          lmmono8-regular.otf
lmmono9regular          lmmono9regular          lmmono9-regular.otf
lmmonocaps10regular     lmmonocaps10regular     lmmonocaps10-regular.otf
lmmonolt10regular       lmmonolt10regular       lmmonolt10-regular.otf
```

```
lmmonoltcond10regular   lmmonoltcond10regular   lmmonoltcond10-regular.otf
lmmonoprop10regular     lmmonoprop10regular     lmmonoprop10-regular.otf
lmmonoproplt10regular   lmmonoproplt10regular   lmmonoproplt10-regular.otf
lmmonoslant10regular    lmmonoslant10regular    lmmonoslant10-regular.otf
```

It does not show from this list but with name lookups first OpenType fonts are checked and then Type1. In this case there are Type1 variants as well but they are ignored. Fonts are registered under all names that make sense and can be derived from their description. So:

```
mtxrun --script font --list --method=name --pattern="latinmodern*mono" --all
```

will give:

```
latinmodernmono                       lmmono9regular              lmmono9-regular.otf
latinmodernmonocaps                   lmmonocaps10oblique         lmmonocaps10-oblique.otf
latinmodernmonocapsitalic             lmmonocaps10oblique         lmmonocaps10-oblique.otf
latinmodernmonocapsnormal             lmmonocaps10oblique         lmmonocaps10-oblique.otf
latinmodernmonolight                  lmmonolt10regular           lmmonolt10-regular.otf
latinmodernmonolightbold              lmmonolt10boldoblique       lmmonolt10-boldoblique.otf
latinmodernmonolightbolditalic        lmmonolt10boldoblique       lmmonolt10-boldoblique.otf
latinmodernmonolightcond              lmmonoltcond10oblique       lmmonoltcond10-oblique.otf
latinmodernmonolightconditalic        lmmonoltcond10oblique       lmmonoltcond10-oblique.otf
latinmodernmonolightcondlight         lmmonoltcond10oblique       lmmonoltcond10-oblique.otf
latinmodernmonolightitalic            lmmonolt10oblique           lmmonolt10-oblique.otf
latinmodernmonolightlight             lmmonolt10regular           lmmonolt10-regular.otf
latinmodernmononormal                 lmmono9regular              lmmono9-regular.otf
latinmodernmonoprop                   lmmonoprop10oblique         lmmonoprop10-oblique.otf
latinmodernmonopropitalic             lmmonoprop10oblique         lmmonoprop10-oblique.otf
latinmodernmonoproplight              lmmonoproplt10oblique       lmmonoproplt10-oblique.otf
latinmodernmonoproplightbold          lmmonoproplt10boldoblique   lmmonoproplt10-boldoblique.otf
latinmodernmonoproplightbolditalic    lmmonoproplt10boldoblique   lmmonoproplt10-boldoblique.otf
latinmodernmonoproplightitalic        lmmonoproplt10oblique       lmmonoproplt10-oblique.otf
latinmodernmonoproplightlight         lmmonoproplt10oblique       lmmonoproplt10-oblique.otf
latinmodernmonopropnormal             lmmonoprop10oblique         lmmonoprop10-oblique.otf
latinmodernmonoslanted                lmmonoslant10regular        lmmonoslant10-regular.otf
latinmodernmonoslantednormal          lmmonoslant10regular        lmmonoslant10-regular.otf
```

Watch the 9 point version in this list. It happens that there are 9, 10 and 12 point regular variants but all those extras come in 10 point only. So we get a mix and if you want a specific design size you really have to be more specific. Because one font can be registered with its font name, full name etc. it can show up more than once in the list. You get what you ask for.

With this obscurity you might wonder why names make sense as lookups. One advantage is that you can forget about special characters. Also, Latin Modern with its design sizes is probably the worst case. So, although for most fonts a name like the following will work, for Latin Modern it gives one of the design sizes:

```
\definefont[SomeFont][name:latinmodernmonolightbolditalic]
```

But this is quite okay:

```
\definefont[SomeFont][name:lmmonolt10boldoblique]
```

So, in practice this method will work out as well as the file method but you can best check if you get what you want.

Method 3: spec

We have now arrived at the third method, selecting by means of a specification. This time we take the family name as starting point (although we have some fall-back mechanisms):

```
\definefont[SomeSerif]          [spec:times]
\definefont[SomeSerifBold]      [spec:times-bold]
\definefont[SomeSerifItalic]    [spec:times-italic]
\definefont[SomeSerifBoldItalic][spec:times-bold-italic]
```

The patterns are of the form:

```
spec:name-weight-style-width
spec:name-weight-style
spec:name-style
```

When only the name is used, it actually boils down to:

```
spec:name-normal-normal-normal
```

So, this is also valid:

```
spec:name-normal-italic-normal
spec:name-normal-normal-condensed
```

Again we can consult the database:

```
mtxrun --script font --list --method=spec lmmono-normal-italic --all
```

This prints the following list. The first column is the family name, the fifth column the font name:

```
latinmodernmono       normal   italic   normal   lmmono10italic        lmmono10-italic.otf
latinmodernmonoprop   normal   italic   normal   lmmonoprop10oblique   lmmonoprop10-oblique.otf
lmmono10              normal   italic   normal   lmmono10italic        lmtti10.afm
lmmonoprop10          normal   italic   normal   lmmonoprop10oblique   lmvtto10.afm
lmmonocaps10          normal   italic   normal   lmmonocaps10oblique   lmtcso10.afm
latinmodernmonocaps   normal   italic   normal   lmmonocaps10oblique   lmmonocaps10-oblique.otf
```

Watch the OpenType and Type1 mix. As we're just investigating here, the lookup looks at the font name and not at the family name. At the TEX end you use the family name:

```
\definefont[SomeFont][spec:latinmodernmono-normal-italic-normal]
```

So, we have the following ways to access this font:

```
\definefont[SomeFont][file:lmmono10-italic]
\definefont[SomeFont][file:lmmono10-italic.otf]
\definefont[SomeFont][name:lmmono10italic]
\definefont[SomeFont][spec:latinmodernmono-normal-italic-normal]
```

As OpenType fonts are preferred over Type1 there is not much chance of a mix-up.

As mentioned in the introduction, qualifications are somewhat inconsistent. Among the weight we find: black, bol, bold, demi, demibold, extrabold, heavy, light, medium, mediumbold, regular, semi, semibold, ultra, ultrabold and ultralight. Styles are: ita, ital, italic, roman, regular, reverseoblique, oblique and slanted. Examples of width are: book, cond, condensed, expanded, normal and thin. Finally we have alternatives which can be anything.

When doing a lookup, some normalizations takes place, with the default always being 'normal'. But still the repertoire is large:

```
helveticaneue medium     normal normal    helveticaneuemedium          HelveticaNeue.ttc index:  0
helveticaneue bold       normal condensed helveticaneuecondensedbold   HelveticaNeue.ttc index:  1
helveticaneue black      normal condensed helveticaneuecondensedblack  HelveticaNeue.ttc index:  2
helveticaneue ultralight italic thin      helveticaneueultralightitalic HelveticaNeue.ttc index:  3
helveticaneue ultralight normal thin      helveticaneueultralight      HelveticaNeue.ttc index:  4
helveticaneue light      italic normal    helveticaneuelightitalic     HelveticaNeue.ttc index:  5
helveticaneue light      normal normal    helveticaneuelight           HelveticaNeue.ttc index:  6
helveticaneue bold       italic normal    helveticaneuebolditalic      HelveticaNeue.ttc index:  7
helveticaneue normal     italic normal    helveticaneueitalic          HelveticaNeue.ttc index:  8
helveticaneue bold       normal normal    helveticaneuebold            HelveticaNeue.ttc index:  9
helveticaneue normal     normal normal    helveticaneue                HelveticaNeue.ttc index: 10
helveticaneue normal     normal condensed helveticaneuecondensed       hlc_____.afm   conflict: roman
helveticaneue bold       normal condensed helveticaneueboldcond        hlbc____.afm
helveticaneue black      normal normal    helveticaneueblackcond       hlzc____.afm   conflict: normal
helveticaneue black      normal normal    helveticaneueblack           hlbl____.afm   conflict: normal
helveticaneue normal     normal normal    helveticaneueroman           lt_50259.afm   conflict: regular
```

The font database

In MkIV we use a rather extensive font database which in addition to bare information also contains a couple of hashes. When you use ConTEXt MkIV and install a new font, you have to regenerate the file database. In a next TEX run this will trigger a reload of the font database. Of course you can also force a reload with:

```
mtxrun --script font --reload
```

As a summary we mention a few of the discussed calls of this script:

```
mtxrun --script font --list somename (== --pattern=*somename*)

mtxrun --script font --list --method=name somename
mtxrun --script font --list --method=name --pattern=*somename*

mtxrun --script font --list --method=spec somename
mtxrun --script font --list --method=spec somename-bold-italic
mtxrun --script font --list --method=spec --pattern=*somename*
mtxrun --script font --list --method=spec --filter="fontname=somename"
mtxrun --script font --list --method=spec
                    --filter="familyname=somename,weight=bold,style=italic,width=condensed"

mtxrun --script font --list --method=file somename
mtxrun --script font --list --method=file --pattern=*somename*
```

The lists shown in before depend on what fonts are installed and their version. They might not reflect reality at the time you read this.

Interfacing

Regular users never deal with the font database directly. However, if you write font loading macros yourself, you can access the database from the TEX end. First we show an example of an entry in the database, in this case TeXGyreTermes Regular.

```
{
    designsize = 100,
    familyname = "texgyretermes",
    filename = "texgyretermes-regular.otf",
    fontname = "texgyretermesregular",
    fontweight = "regular",
    format = "otf",
    fullname = "texgyretermesregular",
    maxsize = 200,
    minsize = 50,
    rawname = "TeXGyreTermes-Regular",
    style = "normal",
    variant = "",
    weight = "normal",
    width = "normal",
}
```

Another example is Helvetica Neue Italic:

```
{
    designsize = 0,
    familyname = "helveticaneue",
    filename = "HelveticaNeue.ttc",
    fontname = "helveticaneueitalic",
    fontweight = "book",
    format = "ttc",
    fullname = "helveticaneueitalic",
    maxsize = 0,
    minsize = 0,
    rawname = "Helvetica Neue Italic",
    style = "italic",
    subfont = 8,
    variant = "",
    weight = "normal",
    width = "normal",
}
```

As you can see, some fields can be meaningless, like the sizes. As using the low level TEX interface assumes some knowledge, we stick here to an example:

```
\def\TestLookup#1%
  {\dolookupfontbyspec{#1}
   pattern: #1, found: \dolookupnoffound
   \blank
   \dorecurse {\dolookupnoffound} {%
     \recurselevel:~\dolookupgetkeyofindex{fontname}{\recurselevel}%
```

```
    \quad
  }%
  \blank}

\TestLookup{familyname=helveticaneue}
\TestLookup{familyname=helveticaneue,weight=bold}
\TestLookup{familyname=helveticaneue,weight=bold,style=italic}
```

You can use the following commands:

```
\dolookupfontbyspec    {key=value list}
\dolookupnoffound
\dolookupgetkeyofindex {key}{index}
\dolookupgetkey        {key}
```

First you do a lookup. After that there can be one or more matches and you can access the fields of each match. What you do with the information is up to yourself.

A few remarks

The fact that modern TEX engines can access system fonts is promoted as a virtue. The previous sections demonstrated that in practice this does not really free us from a name mess. Of course, when we use a really small TEX tree, and system fonts only, there is not much that can go wrong, but when you have extra fonts installed there can be clashes.

We're better off with file names than we were in former times when operating systems and media forced distributors to stick to 8 characters in file names. But that does not guarantee that today's shipments are more consistent. And as there are still some limitations in the length of font names, obscure names will be with us for a long time to come.

Hans Hagen

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

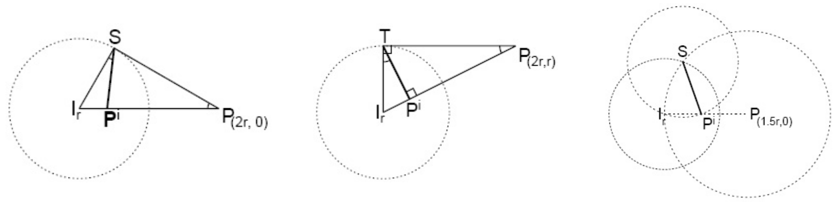
Professional software for graphics (and animation as well) is the commercial Mathematica. A free Mathematica reader can be downloaded, however, which ensures that one can view and animate work of others, the so-called mathematica notebooks, to start with Mathematica's own demos: a mer à boire.
See http://mathworld.wolfram.com/Inversion.html 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 $|\mathrm{IP}^i|\cdot|\mathrm{IP}| = \mathrm{r}^2$. Explicitly

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

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$^{\text{nd}}$ 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
```

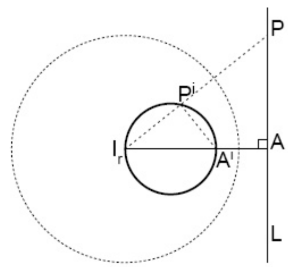
Line inversion

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.

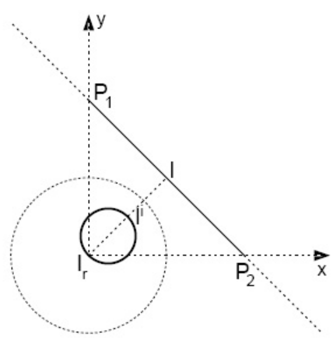


Let A be the point where the perpendicular meets L, and let A^i be the inverse point of A. Mark any point P on L, and let P^i be its inverse point. Since $|IA^i| \cdot |IA| = |IP^i| \cdot IP| = r^2$, it follows that

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

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]

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

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.

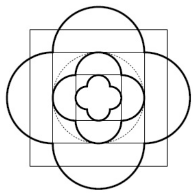
In my earlier MetaFont version I used the equation solving possibilities of MetaFont by specifying equations for the footpoint of the perpendicular from I on the line P_1P_2. The use of rotation is simpler and elegant.

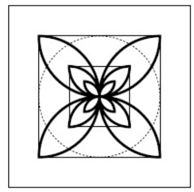
```
/lineinversion
% x1 y1 x2 y2: the points which determine the line
% mx my r:     the centre and radius of the inversion circle
%==>
% mix miy ri:  centre and radius of the inverted line
{0 begin
/eps .0001 def /r exch def
/my exch def /mx exch def
/y2 exch def /x2 exch def
/y1 exch def /x1 exch def
/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.





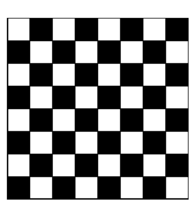
We could extend the above to concentric pentagons, hexagons, but … I don't expect new beautiful results: instead of 4 circular arcs we will obtain 5, 6. . . .

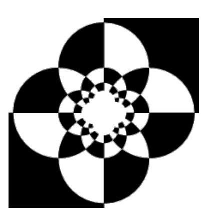
The example is a test for `linepieceinversion`, though it does not exercise the shift of origin.

```
/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
```

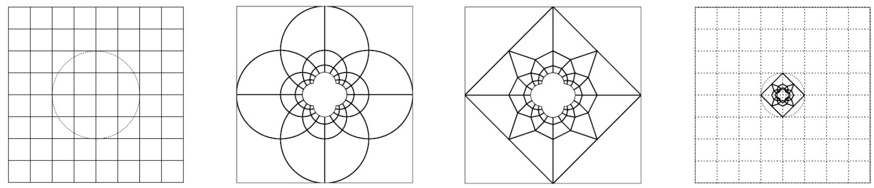
It was troublesome to draw the correct part of the circle without clipping.

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.



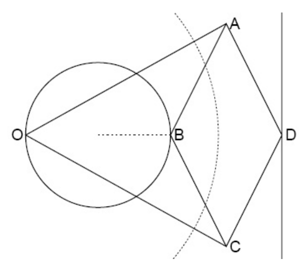


Variation of the inversion of a chessboard is a stainless glass impression of the 'inverse' of a grid of 64 squares, as displayed below. In the inversions the dashed surrounding square indicates ∞, the inverse of the centre.



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.



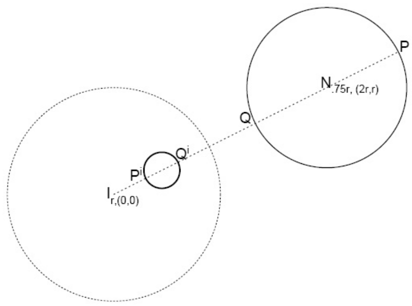
A disadvantage of this apparatus is that only a to and fro movement along a circular arc is obtained. Transformation of to and fro motion into rotation is realized in a car (think of pistons) via a crank-shaft or a totter system.

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

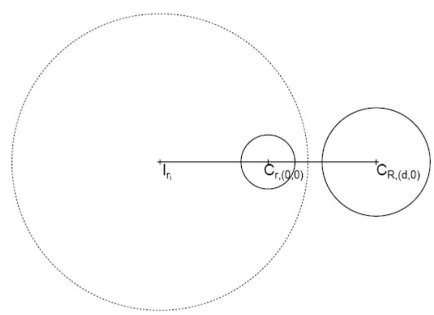
```
/circleinversion
% Nx, Ny, R: centre and radius of the to be inverted circle
% Ix, Iy, r: centre and  radius of the inversion circle
%==>
% xi,yi, ri: centre and radius of the inverted circle
{0 begin
/r exch def  /y exch def /x exch def
/R exch def  /Y exch def /X exch def
/Xmx X x sub def
/Ymy 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
```

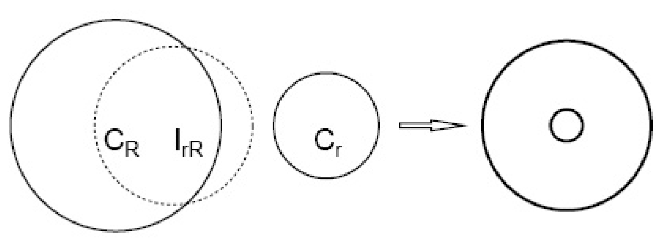
Applications

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.

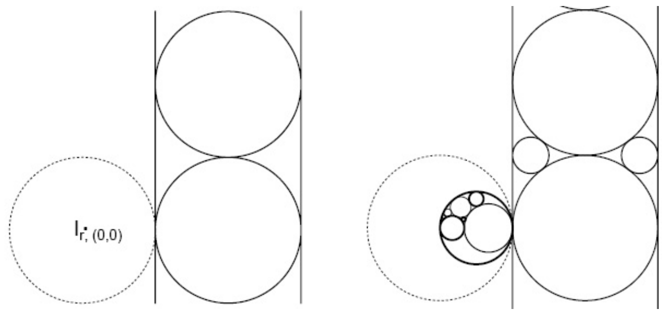
PS code The library operators can be found elsewhere in this note.

```
%!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.

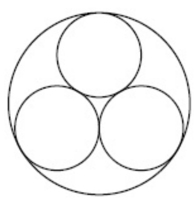


If we invert the stacked circles and bordering verticals in I_r, see the left picture, then we arrive at the picture at the right. Well … not quite: two small circles have been added. In the right picture the left vertical is converted into 'the main' circle which is covered by the other inverted circles.

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.



$$r = \langle\mathit{parameter}\rangle\ x = 0\ y = r_i + r \quad \text{(upper inner circle)}$$

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

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

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

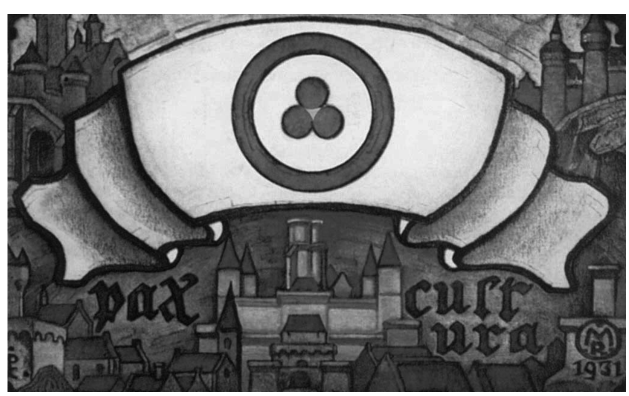
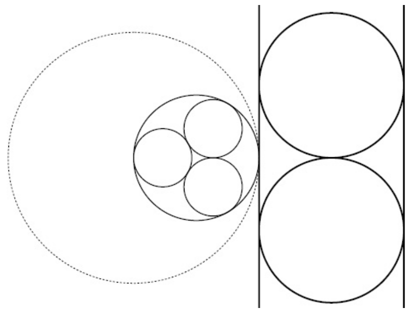
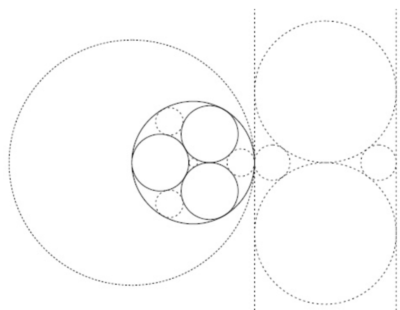


Figure 1. Rerich's *Pax Cultura*

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.

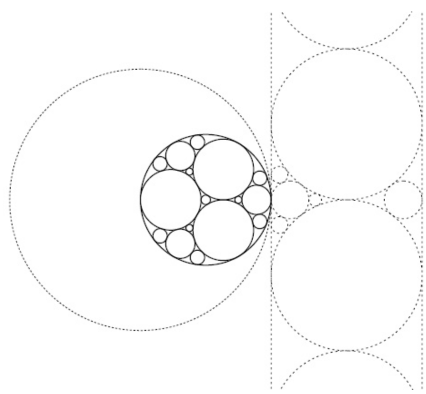


In the transformed figure the radius (and centre) of the small circle can be obtained as .25r^i, where r^i is the radius of the inverted circles C_r^i. The middle circle is added in the Trinity symbol, and the second level Перих is obtained via backtransformation and a rotation or two in the original 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



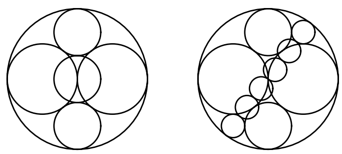
By Soddy's formula which gives a relation between the radii of touching circles, and goes back to Descartes' circle theorem

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

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.



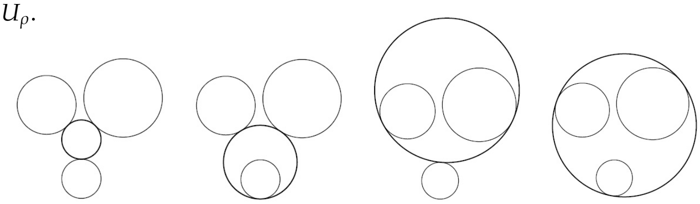
Interesting visual result of some relations between the radii of mutual tangent circles.

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 2$^{\text{nd}}$ and 3$^{\text{rd}}$ picture above also form a conjugate pair.

Calculation 1$^{\text{st}}$ and 2$^{\text{nd}}$ 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)

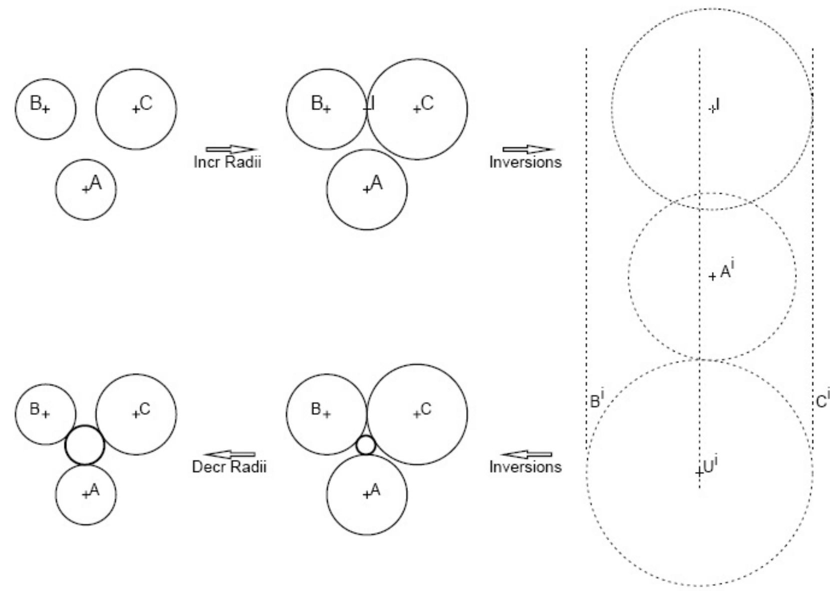


Figure 2. Inscribed circle

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.

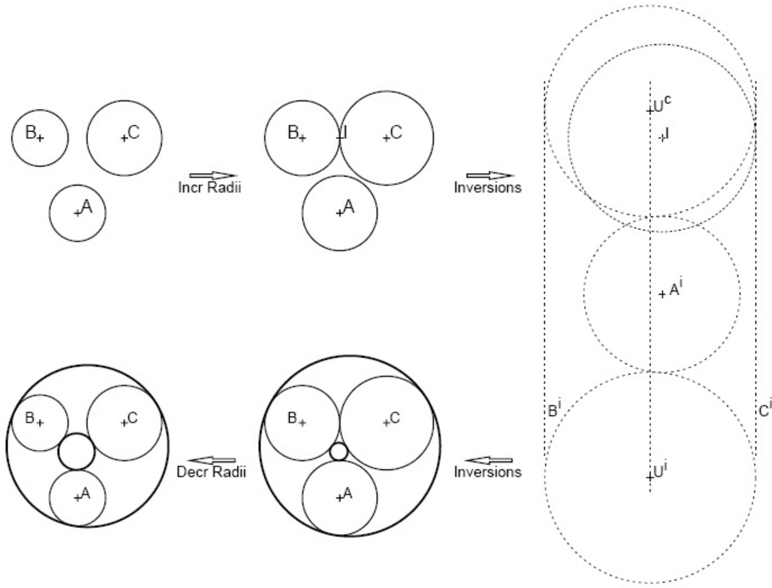


Figure 3. In- and circumscribed circle

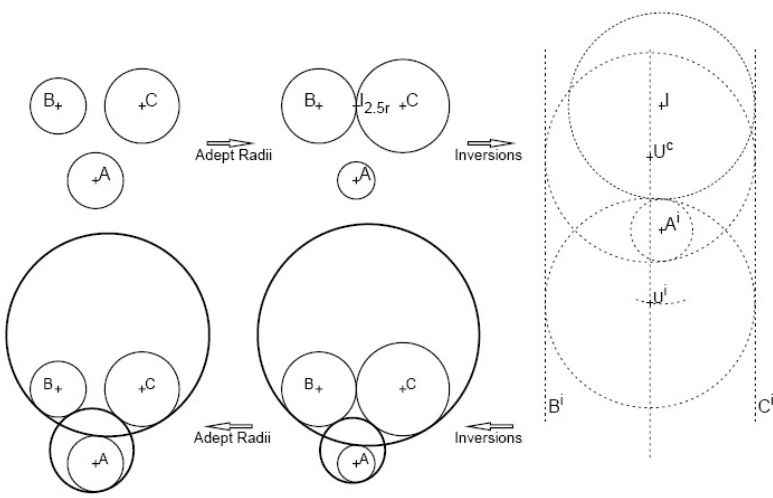


Figure 4. Two circles inside and one outside and vice versa

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

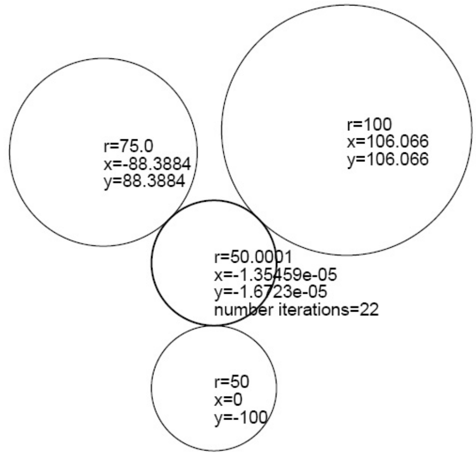
$$\begin{pmatrix} x_{21} & y_{21} \\ x_{32} & y_{32} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \mathrm{f}_r(21) \\ \mathrm{f}_r(32) \end{pmatrix}$$

where

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

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

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

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 3$^{\text{rd}}$ 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.

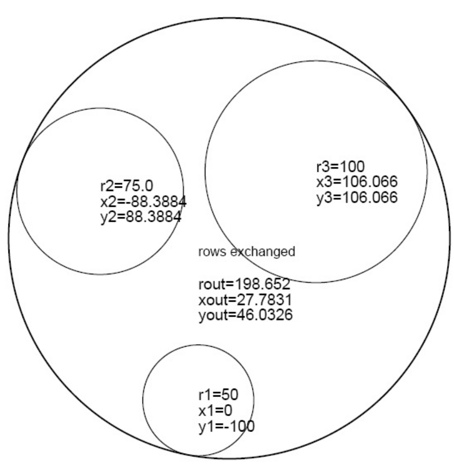
Auxiliaries

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

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

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

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

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

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 3$^{\text{rd}}$ 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.

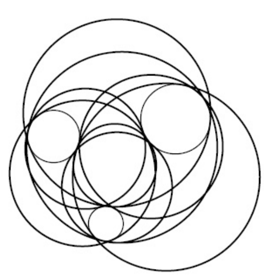
Use of the `inscribed` operator, where x_i y_i r_i denote the centre and radius of the calculated inscribed circle, is done by the invoke

```
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

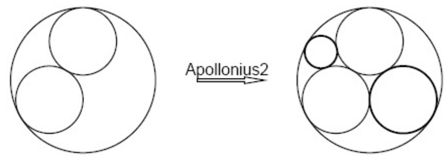
- the inscribed circle, with an invoke similar to the above with `inscribed` changed by `Apollonius`
- the circumscribed circle, with the invoke with the negative radii
  ```
  x1 y1 r1 neg
  x2 y2 r2 neg
  x3 y3 r3 neg Apollonius
               /rcircum exch def /ycircum exch def /xcircum exch def
  ```
- 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

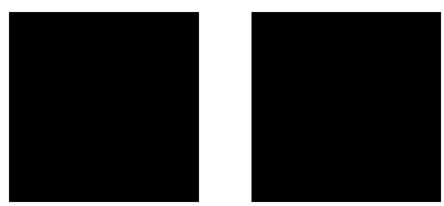
So far I qualified the second root as a spurious solution of the quadratic equation in r. In the case of this section we do have two solutions, so the second value for r, together with its centre, are also delivered by `Apollonius2`.



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.

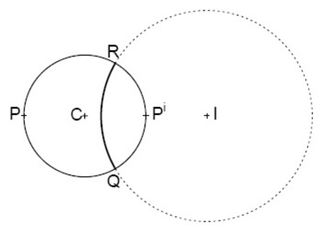
Applications [26]

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 $\widehat{RQ}$, 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.



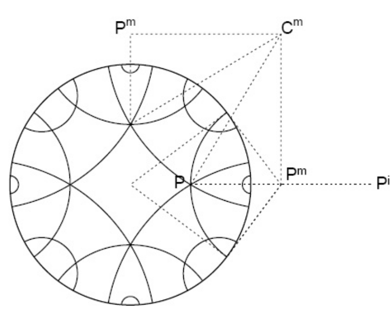
Escher's grids

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 $\approx$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.

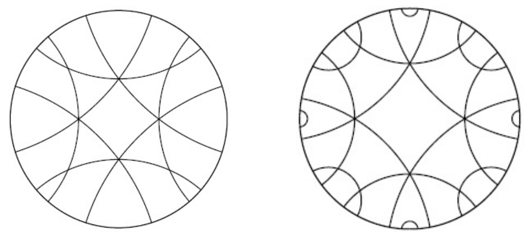
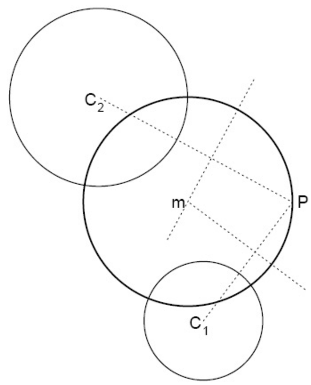


Figure 5. Hyperbolic Escher-like grid

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

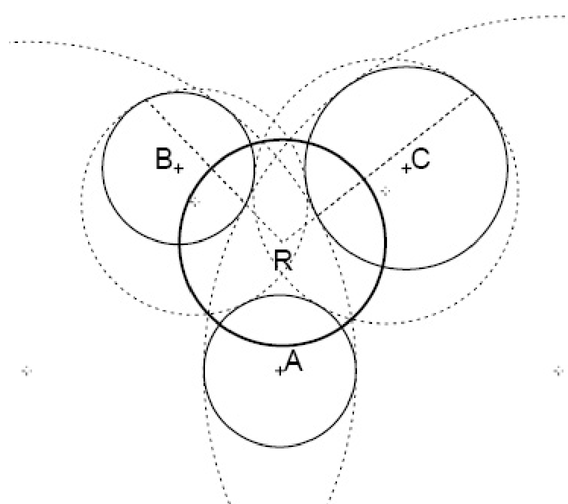
```
%!PS Orthogonal arcs through one point within a circle. CGL March 2010
%%BoundingBox: 0 0 620 790

(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.

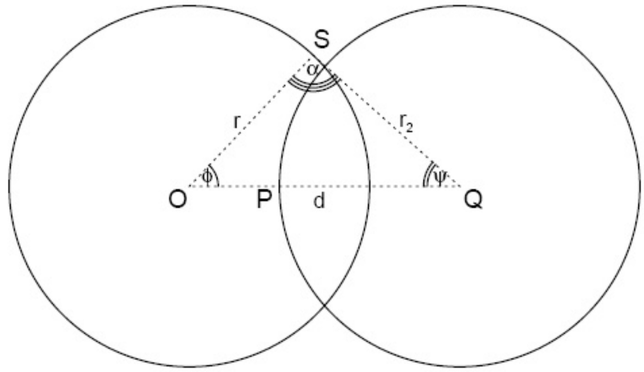
Inversion in the radical circle leaves the given circles unchanged,[27] but transforms the two conjugate solution circles into one another. Under the same inversion, the corresponding points of tangency of the two solution circles are transformed into one another.[28] Hence, the lines connecting these 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.

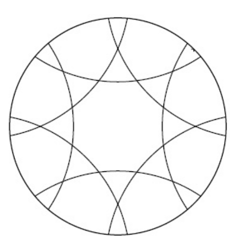
We can simplify, however, by subtracting equation 2 from 1, as done earlier in this note, and obtain

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

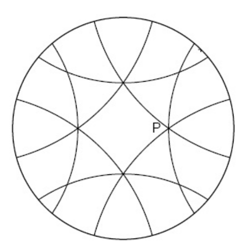
Substitution of this value in the equation for $C_{r,(0,0)}$, yields for the ordinate

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

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?

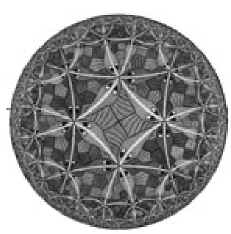
In the implementation the user is asked for an (estimate of the) upper bound of the radius of the wanted circle. As lower bound I assumed that the centre of the wanted circle is at the boundary of the main circle. This limits the use of `circlesatalpha`.

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.

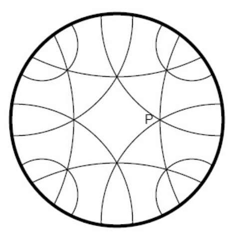
Remark2 I was on the wrong track, too much fixed at an angle of 80°. Coxeter's results of 1996 are by far superior, but … the used math, especially `The Biquadratic field` $\mathbb{Q}(\sqrt{2}+\sqrt{3})$, and in general the used hyperbolic geometry, I'm not yet familiar with, alas. Nevertheless … it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

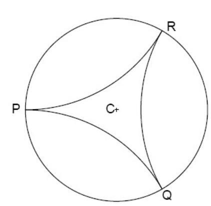


Coxeter's grid

Circle covered by triangles

Let us choose points P, Q and R equally distributed along the circumference of a circle. Draw the orthogonal arcs $\widehat{PQ}$, $\widehat{QR}$ and $\widehat{RP}$. 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.

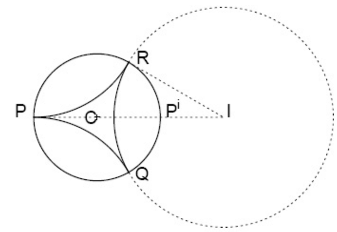
The triangle is also known under the name hypocycloid, which for the case of the regular $\triangle$ is obtained after rolling a circle with radius $\frac{1}{3}$R along the inside of the main circle with radius R. The triangle has sum 0 of the inner angles.



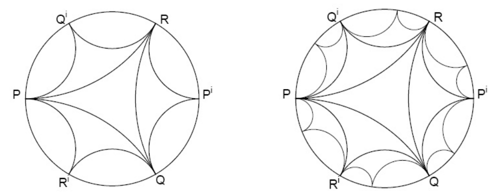
Interesting pictures are obtained when we invert the triangle repeatedly in its sides.

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 $\widehat{RQ}$ is an arc. Then $|\overrightarrow{IP}| \cdot |\overrightarrow{IP^i}| = r^2$, because the circles intersect orthogonally, i.e. $\overrightarrow{IR}$ is tangent to the clipping circle C.

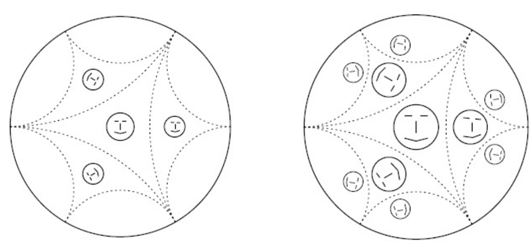


After inversion of P, Q and R, and drawing the (orthogonal) circular arcs, PQ^i, Q^iR et cetera, the clipping circle C is covered by four hyperbolically congruent triangles: PQR, and its three hyperbolically mirrored images PRQ^i, PQR^i, and RQP^i.



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$^{\text{nd}}$ level inversions, the operators `pointinversion` and `circleinversion` are invoked repeatedly: the invoke for the 1$^{\text{st}}$ level inversion is immediately followed by the invoke for the 2$^{\text{nd}}$ level inversion in the procedures for `pathforall`.

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

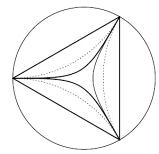
```
%!PS-Adobe-EPSF-3.0  pathforall use, cgl jan 2010
%%BoundingBox: 150 250 250 350

(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

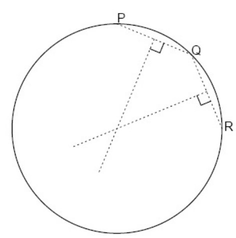
A circle is usually defined by the set of points which lie at a constant distance from the circle centre, usually the origin.

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

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.

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

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.

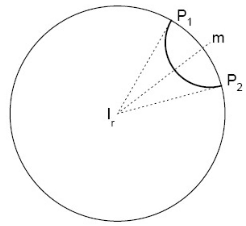


Below an operator is given for drawing a circle given three points on the circumference.[33]

```
/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
```

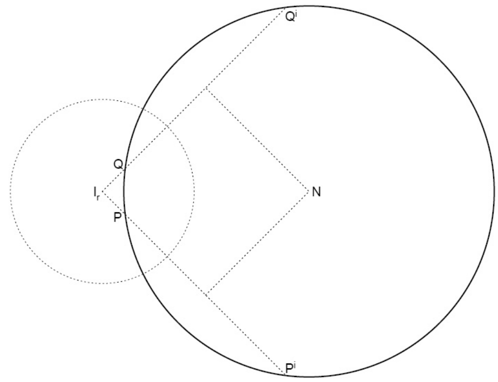
I needed the above for drawing orthogonal circles through two points on the circumference of a circle.

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 $\overrightarrow{PP^i}$ and $\overrightarrow{QQ^i}$ 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
```

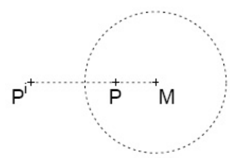
I thought that I needed this operator for drawing Escher's Circle Limit grids. Closer inspection of Escher's Circle Limit III grid yielded that only one point within the circle is sufficient for drawing the grid. The same holds for Circle Limit I, although I'm puzzled about which grid he used, and why.

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 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.

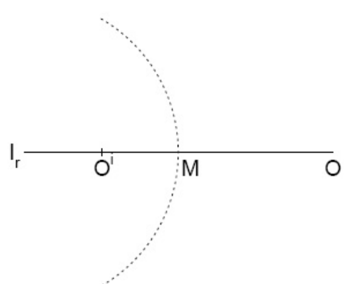
Reflection is not inversion

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 Bogusłav 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 $\vec{\mathrm{P}}$ and $\vec{\mathrm{P}}^i$. 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.

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

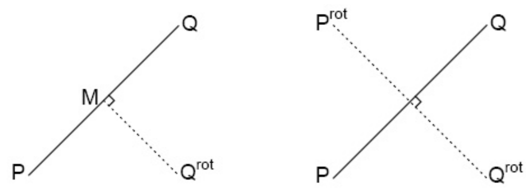
```
/size % x y ==> sqrt(x^2+y^2)
      % not robust against 0 0 on the stack
{abs dup  3 -1 roll abs dup 3 1 roll % |y| |x| |y| |x|
le {size}                            % |y|<=|x|
   {exch dup 3 1 roll                % |y| |x| |y|
     div                             % |y| |x|/|y|
     dup  mul 1 add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

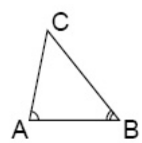
```
/ortho
%lx ly   point on left leg
%x y     cornerpoint
%s       size s of marker
%==>
%ortho symbol drawn of size s
{0 begin
 /s  exch def
 /y  exch def /x  exch def
 /ly exch def /lx exch def
 gsave x y translate
 lx x sub ly y sub atan neg  rotate
 0 s moveto s s lineto s 0 lineto stroke
 grestore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

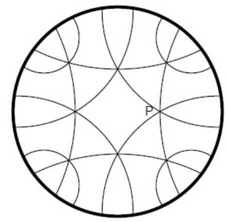
Binary tree The difference with the code in the EuroTEX2009 proceedings is the careful use of `currentpoint`, which resulted in a concise code.

```
/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.

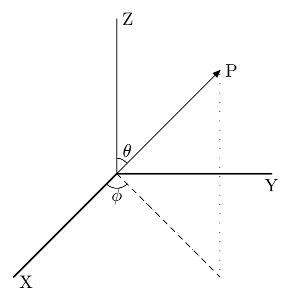
```
/Hfractal{/k exch def
   gsave draw
   /k k 2 mul 3 div def
   k 1 gt { 90 rotate k Hfractal
            -180 rotate k Hfractal}if
   /k k 3 mul 2 div def
   grestore}def
/draw{0 k rlineto
    currentpoint stroke translate
    0 0 moveto}def
```

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
```

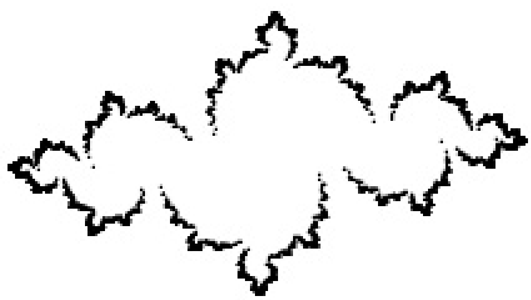
Indispensable. A practical variant with fixed viewing angles, `ptpf`, is part of the library too.

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.



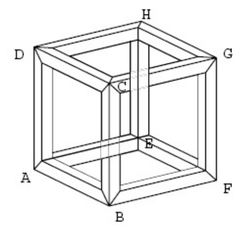
```
%!PS-Adobe- Julia sets, cgl May 97
%%Author: Peitgen e.a. (1992): Chaos and fractals. Springer-Verlag
%%BoundingBox: [-100 0 100 50]
300 300 translate
/Courier findfont 7 scalefont setfont
/s 50 def                      % scaling
/cr -1 def    /ci  0 def       %c as complex number
/cr  0 def    /ci  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
```

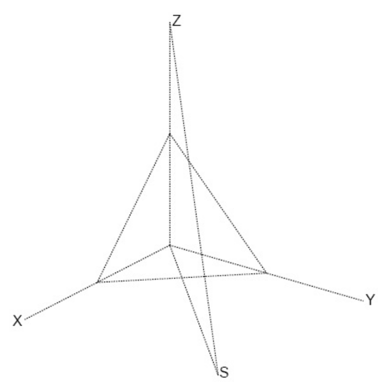
Intersection of planes `solve33` can be used to calculate the intersection point of the planes specified in matrix notation, for example as

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

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
```

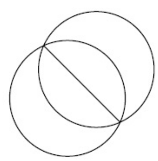
```
  /x13 exch def /x12 exch def /x11 exch def /rh1 exch def
  %calculation determinant
  /determinant x11 x22 x33 mul x23 x32 mul sub mul
               x12 x21 x33 mul x23 x31 mul sub mul sub
               x13 x21 x32 mul x22 x31 mul sub mul add def
%elimination last column, bottom up, to keep x,y 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.

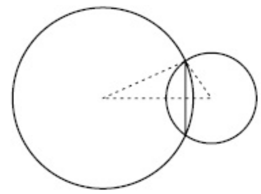
I needed this operator to determine the intersection of two grid lines in Escher's Circle Limit I.



```
/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)

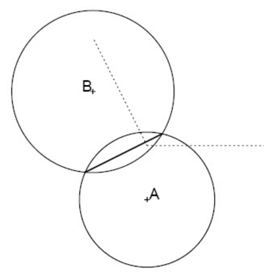
$$h = \frac{2}{d}\sqrt{s(s-d)(s-r1)(s-r2)} \qquad \text{with} \qquad s = \frac{r1 + r2 + d}{2}.$$



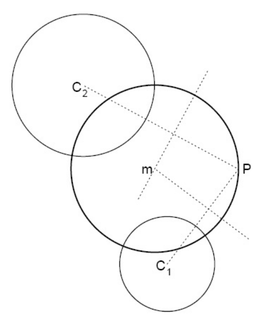
No need for solving quadratic equations. I thought that I needed this operator for finding the radical circle of three circles, not so because in that way it is an ill-posed problem. The operator maybe useful, nonetheless.

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

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

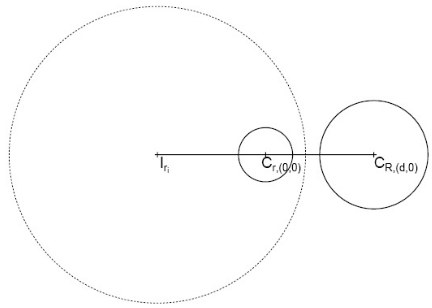


Circle orthogonal to two circles and passes through P The centre of the circle, m, is the intersection of the middle perpendiculars of P and the inverse points towards C_1 and C_2.

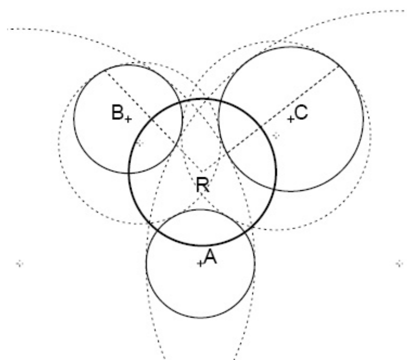


Inversion circle to conjugate circles, special case The problem is: given two distinct circles find the inversion circle which transforms the circles in each other.

```
/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`.

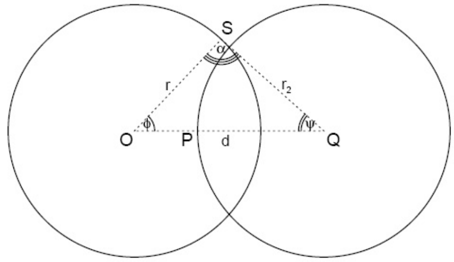


Interesting is that tangent point of two touching circles have to be determined as an intermediate step, which is an ill-posed problem if done by calculating the intersection of the two circles. I don't know of a situation where an ill-posed problem can so gracefully be circumvented. In this case it 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.

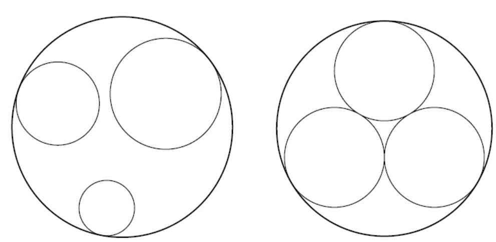
An ill-posed problem is not to be confused with an ill-conditioned solution technique. For example in solving a system of linear equations the complete or partial pivoting strategy is a much better conditioned numerical method that just Gaussian elimination, for the same problem.

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.

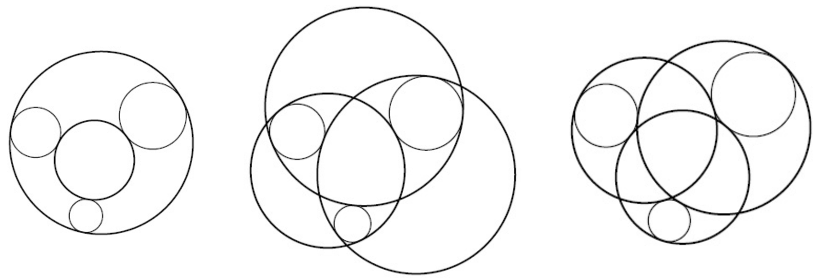


```
/circumscribed
%Purpose: Calculate circumscribed circle given three disjunct circles
% x1 y1 r1, x2 y2 r2, x3 y3 r3: centres and radii of three (disjunt) circles
%==>
% x y r:                       midpoint and radius of the inscribed circle
{0 begin
/r3 exch def /y3 exch def /x3 exch def
```

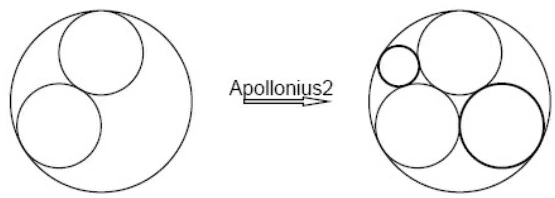
```
/r2 exch def /y2 exch def /x2 exch def
/r1 exch def /y1 exch def /x1 exch def
%auxiliary data
/x21 x2 x1 sub def /x31 x3 x1 sub def  /x32 x3 x2 sub def
/y21 y2 y1 sub def /y31 y3 y1 sub def  /y32 y3 y2 sub def
/r21 r2 r1 sub def /r31 r3 r1 sub def  /r32 r3 r2 sub def
/mx21 x2 x1 add 2 div def /mx31 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.

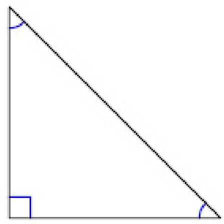
Appendix II: use of the PostScript library in MetaPost

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

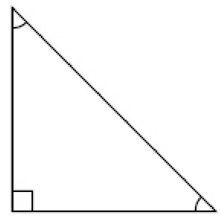
MetaPost with included PostScript and library use

```
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^{i}P 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.

EuroTEX 2010

The Italian TEX User Group (GuIT) is very proud to invite you to EuroTEX 2010. The conference will be held from 25 to 29 August at Sant'Anna School of Advanced Studies in Pisa, Italy.

Further information on the registration, programme, accommodation, and social events will be soon available on our website:

http://www.guit.sssup.it/eurotex2010/eurotex.en.php

We hope to see you soon in Pisa.

Grouping in hybrid environments

Keywords
ConTeXt mkiv, luatex, grouping, underbar, overbar, overstrike, text backgrounds

Variants

After using TEX for a while you get accustomed to one of its interesting concepts: grouping. Programming languages like Pascal and Modula have keywords `begin` and `end`. So, one can say:

```
if test then begin
    print_bold("test 1")
    print_bold("test 2")
end
```

Other languages provide a syntax like:

```
if test {
    print_bold("test 1")
    print_bold("test 2")
}
```

So, in those languages the `begin` and `end` and/or the curly braces define a 'group' of statements. In TEX on the other hand we have:

```
test \begingroup \bf test \endgroup test
```

Here the second `test` comes out in a bold font and the switch to bold (basically a different font is selected) is reverted after the group is closed. So, in TEX grouping deals with scope and not with grouping things together.

It depends on the language whether locally defined variables are visible afterwards. In languages like Lua we have constructs like:

```
for i=1,100 do
    local j = i + 20
    ...
end
```

Here `j` is visible after the loop ends unless prefixed by `local`. Yet another example is MetaPost:

```
begingroup ;
    save n ; numeric n ; n := 10 ;
    ...
endgroup ;
```

Here all variables are global unless they are explicitly saved inside a group. This makes perfect sense as the resulting graphic also has a global (accumulated) property. In practice one will rarely need grouping, contrary to TEX where one really wants to keep changes local, if only because document content is so unpredictable that one never knows when some change in state happens.

But in TEX variables are local unless a `\global` prefix (or one of the shortcuts) is used.

In principle it is possible to carry over information across a group boundary. Consider this somewhat unrealistic example:

```
\begingroup
    \leftskip 10pt
    \begingroup
        ....
        \advance\leftskip 10pt
        ....
    \endgroup
\endgroup
```

How to carry the increased `\leftskip` over the group boundary without using a global assignment which could have more drastic side effects? Here is the trick:

```
\begingroup
    \leftskip 10pt
    \begingroup
        ....
        \advance\leftskip 10pt
        ....
        \expandafter
    \endgroup
    \expandafter \leftskip \the\leftskip
\endgroup
```

This is a typical example of the kind of code that gives new users the creeps but normally they never have to do that kind of coding. Also, this kind of trick assumes that one knows how many groups are involved.

Implication

What does this all have to do with LuaTeX and MkIV? The user interface of ConTeXt provide lots of commands like:

```
\setupthis[style=bold]
\setupthat[color=green]
```

Most of them obey grouping. However, consider a situation where we use Lua code to deal with some aspect of typesetting, for instance numbering lines or adding ornamental elements to the text. In ConTeXt we flag such actions with attributes and often the real action takes place a bit later, for instance when a paragraph or page becomes available.

A comparable pure TeX example is the following:

```
{test test \bf test \leftskip10pt test}
```

Here the switch to bold happens as expected but no `\leftskip` of 10pt is applied. This is because the set value is already forgotten when the paragraph is actually typeset. So in fact we would need:

```
{test test \bf test \leftskip10pt test \par}
```

Now, say that we have:

```
{test test test \setupflag[option=1]
                \flagnexttext test}
```

We flag some text (using an attribute) and expect it to get a treatment where option 1 is used. However, the real action might take place when TeX deals with the paragraph or page and by that time the specific option is already forgotten or it might have received another value. So, the rather natural TeX grouping does not work out that well in a hybrid situation.

As the user interface assumes a consistent behaviour we cannot simply make these settings global even if this makes much sense in practice. One solution is to carry the information with the flagged text i.e. associate it somehow with the attribute's value. Of course, as we never know in advance when this information is used, this might result in quite some states being stored persistently.

A side effect of this 'problem' is that new commands might get suboptimal user interfaces (especially inheritance or cloning of constructs) that are somewhat driven by these 'limitations'. Of course we may wonder if the end user will notice this.

To summarize this far, we have three sorts of grouping to deal with:

- TeX's normal grouping model limits its scope to the local situation and normally has only direct and local consequences. We cannot carry information over groups.
- Some of TeX's properties are applied later, for instance when a paragraph or page is typeset and in order to make 'local' changes effective, the user needs to add explicit paragraph ending commands (like `\par` or `\page`).
- Features dealt with asynchronously by Lua are at that time unaware of grouping and variables set that were active at the time the feature was triggered so there we need to make sure that our settings travel with the feature. There is not much that a user can do about it as this kind of management has to be done by the feature itself.

It is the third case that I will give an example of in the next section. I will leave it up to the user whether it gets noticed in the user interface.

An example

A group of commands that has been reimplemented using a hybrid solution is underlining or more generically: bars. Just take a look at the following examples and try to get an idea of how to deal with grouping. Keep in mind that:

- Colors are attributes and are resolved in the back-end, so way after the paragraph has been typesetting.
- Overstrike is also handled by an attribute and gets applied in the back-end as well, before colours are applied.
- Nested overstrikes might have different settings.
- An overstrike rule either inherits from the text or has its own colour setting.

First an example where we inherit colour from the text:

```
\definecolor[myblue][b=.75]
\definebar[myoverstrike][overstrike][color=]

Test \myoverstrike{%
    Test \myoverstrike{\myblue
        Test \myoverstrike{Test}
        Test}
    Test}
Test
```

Test ~~Test~~ ~~Test~~ ~~Test~~ ~~Test~~ ~~Test~~ Test

Because colour is also implemented using attributes and processed later we can access that information when we deal with the bar.

The following example has its own colour setting:

```
\definecolor[myblue][b=.75]
\definecolor[myred] [r=.75]
\definebar[myoverstrike][overstrike][color=myred]

Test \myoverstrike{%
    Test \myoverstrike{\myblue
        Test \myoverstrike{Test}
        Test}
    Test}
Test
```

Test ~~Test~~ ~~Test~~ ~~Test~~ ~~Test~~ ~~Test~~ Test

See how can we colour the levels differently:

```
\definecolor[myblue] [b=.75]
\definecolor[myred]  [r=.75]
\definecolor[mygreen][g=.75]

\definebar[myoverstrike:1]
          [overstrike][color=myblue]
\definebar[myoverstrike:2]
          [overstrike][color=myred]
\definebar[myoverstrike:3]
          [overstrike][color=mygreen]

Test \myoverstrike{%
    Test \myoverstrike{%
        Test \myoverstrike{Test}
        Test}
    Test}
Test
```

Test ~~Test~~ ~~Test~~ ~~Test~~ ~~Test~~ ~~Test~~ Test

Watch this:

```
\definecolor[myblue] [b=.75]
\definecolor[myred]  [r=.75]
\definecolor[mygreen][g=.75]

\definebar[myoverstrike]
          [overstrike][max=1,dy=0,offset=.5]
\definebar[myoverstrike:1]
          [myoverstrike][color=myblue]
\definebar[myoverstrike:2]
          [myoverstrike][color=myred]
\definebar[myoverstrike:3]
          [myoverstrike][color=mygreen]

Test \myoverstrike{%
    Test \myoverstrike{%
        Test \myoverstrike{Test}
        Test}
    Test}
Test
```

Test ~~Test~~ ~~Test~~ ~~Test~~ ~~Test~~ ~~Test~~ Test

Is this the perfect user interface? Probably not, but at least it keeps the implementation quite simple.

The behaviour of the MkIV implementation is roughly the same as in MkII, although now we specify the dimensions and placement in terms of the ratio of the x-height of the current font.

```
Test \overstrike{Test \overstrike{Test
  \overstrike{Test} Test} Test} Test \blank
Test \underbar  {Test \underbar  {Test
  \underbar  {Test} Test} Test} Test \blank
Test \overbar   {Test \overbar   {Test
  \overbar   {Test} Test} Test} Test \blank
Test \underbar  {Test \overbar   {Test
  \overstrike{Test} Test} Test} Test \blank
```

Test ~~Test~~ ~~Test~~ ~~Test~~ ~~Test~~ ~~Test~~ Test

Test Test Test Test Test Test Test

Test Test Test Test Test Test Test

Test Test Test ~~Test~~ Test Test Test

As a bonus this mechanism can also provide simple backgrounds. The normal background mechanism uses MetaPost and the advantage is that we can use arbitrary shapes but it also carries some limitations. When the development of LuaTEX is a bit further along the road I will add the possibility to use MetaPost shapes in this mechanism.

Before we come to backgrounds, first take a look at these examples:

```
\startbar[underbar]  \input zapf \stopbar \blank
\startbar[underbars] \input zapf \stopbar \blank
```

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

First notice that it is no problem to span multiple lines and that hyphenation is not influenced at all. Second you can see that continuous rules are also possible. From such a continuous rule to a background is a small step:

```
\definebar
  [backbar]
  [offset=1.5,rulethickness=2.8,color=blue,
   continue=yes,order=background]

\definebar
  [forebar]
  [offset=1.5,rulethickness=2.8,color=blue,
   continue=yes,order=foreground]
```

The following example code looks messy but this has to do with the fact that we want properly spaced sample injection.

```
from here
    \startcolor[white]%
        \startbar[backbar]%
            \input zapf
            \removeunwantedspaces
        \stopbar
    \stopcolor
\space till here
\blank
from here
    \startbar[forebar]%
        \input zapf
        \removeunwantedspaces
    \stopbar
\space till here
```

from here Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on. till here

from here [redacted] till here

Watch how we can use the order to hide content. By default rules are drawn on top of the text.

Nice effects can be accomplished with transparencies:

```
\definecolor [tblue] [b=.5,t=.25,a=1]
\setupbars [backbar] [color=tblue]
\setupbars [forebar] [color=tblue]
```

We use as example:

```
from here {\white \backbar{test test}
    \backbar {nested nested} \backbar{also also}}
till here
from here {\white \backbar{test test
    \backbar {nested nested}          also also}}
till here
from here {\white \backbar{test test
    \backbar {nested nested}          also also}}
till here
```

from here test test nested nested also also till here from here test test nested nested also also till here from here test test nested nested also also till here

The darker nested variant is just the result of two transparent bars on top of each other. We can limit stacking, for instance:

```
\setupbars[backbar][max=1]
\setupbars[forebar][max=1]
```

This gives

from here test test nested nested also also till here from here test test nested nested also also till here from here test test nested nested also also till here

There are currently some limitations, mostly due to the fact that MkIV uses only one attribute for this feature and a change in the value therefore triggers different handling. So, there is no real nesting here.

The default commands are defined as follows:

```
\definebar[overstrike]
```

```
        [method=0,dy= 0.4,offset= 0.5]
\definebar[underbar]
        [method=1,dy=-0.4,offset=-0.3]
\definebar[overbar]
        [method=1,dy= 0.4,offset= 1.8]

\definebar[overstrikes]
        [overstrike] [continue=yes]
\definebar[underbars]
        [underbar]   [continue=yes]
\definebar[overbars]
        [overbar]    [continue=yes]
```

As the implementation is rather non-intrusive you can use bars almost everywhere. You can underbar a whole document but you can stick to fooling around with for instance formulas equally well.

```
\definecolor [tred]   [r=.5,t=.25,a=1]
\definecolor [tgreen] [g=.5,t=.25,a=1]
\definecolor [tblue]  [b=.5,t=.25,a=1]

\definebar [mathred]   [backbar] [color=tred]
\definebar [mathgreen] [backbar] [color=tgreen]
\definebar [mathblue]  [backbar] [color=tblue]

\startformula
\mathred{e} =
  \mathgreen{\white mc} ^ {\mathblue{\white e}}
\stopformula
```

We get:

$$e = mc^{e}$$

We started this chapter with some words on grouping. In the examples you see no difference between adding bars and for instance applying colour. However you need to keep in mind that this is only because behind the screens we keep the current settings along with the attribute. In practice this is only noticeable when you do lots of (local) changes to the settings. Take:

```
{test test test
  \setupbars[color=red] \underbar{test} test}
```

This results in a local change in settings, which in turn will associate a new attribute to `\underbar`. So, in fact the following underbar becomes a different one from the previous underbars. When the page is prepared, the unique attribute value will relate to those settings. Of course there are more mechanisms where such associations take place.

More to come

Is this all there is? No, as usual the underlying mechanisms can be used for other purposes as well. Take for instance in-line notes:

```
According to Wikipedia this is the longest
English word:
pneumonoultramicroscopicsilicovolcanoconiosis~%
\shiftup {other long
words are pseudopseudohypoparathyroidism and
flocci-nauci-nihili-pili-fication}. Of course
in languages like Dutch and German we can make
arbitrary long words by pasting words together.
```

This will produce:

According to Wikipedia this is the longest English word: pneumonoultramicroscopicsilicovolcanoconiosis other long words are pseudopseudohypoparathyroidism and floccinaucinihilipilification. Of course in languages like Dutch and German we can make arbitrary long words by pasting words together.

I wonder when users really start using such features.

Summary

Although under the hood the MkIV bar commands are quite different from their MkII counterparts users probably won't notice much difference at first sight. However, the new implementation does not interfere with the par builder and other mechanisms. Plus, it is configurable and it offers more functionality. However, as it is processed in delayed fashion, side effects might occur that are not foreseen.

So, if you ever notice such unexpected side effects, you know where it might result from: what you asked for is processed much later and by then the circumstances might have changed. If you suspect that it relates to grouping there is a simple remedy: define a new bar command in the document preamble instead of changing properties mid-document. After all, you are supposed to separate rendering and content in the first place.

Hans Hagen

4th ConTeXt Meeting

September 13–18, 2010
Brejlov (Prague), Czech Republic

Meeting
- meet new TeX friends, present your results and ideas
- get help from the experienced users
- get in touch with the latest developement of ConTeXt and LuaTeX
- Monday evening to Saturday morning


Place
- Mill *Brejlov*: a place to work & rest, http://www.brejlov.cz
- on the bank of *Sázava* river, beautiful countryside
- 30 km southeast of *Prague* (near *Týnec nad Sázavou*)
- enjoy swimming in the river, canoeing, walking, or cycling
- taste Czech cuisine, beer & wine
- visit *Prague* on the weekend before or after the meeting


See you in Brejlov!

http://meeting.contextgarden.net/2010

CSTUG

OpenType PostScript fonts with unusual units-per-em values

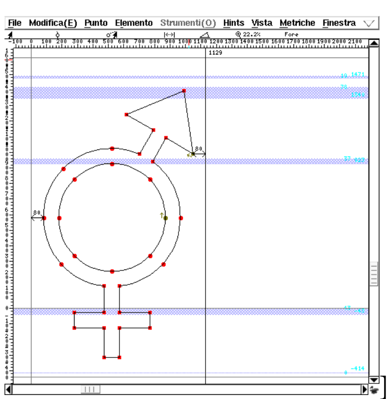
Abstract
OpenType fonts with Postscript outline are usually defined in a dimensionless workspace of 1000×1000 units per em (upm). Adobe Reader exhibits a strange behaviour with pdf documents that embed an OpenType PostScript font with unusual upm: this paper describes a solution implemented by LuaTEX that resolves this problem.

Keywords
LuaTeX, ConTeXt Mark IV, OpenType, FontMatrix.

Introduction

Opentype is a font format that encompasses three kinds of widely used fonts:
1. outline fonts with cubic Bézier curves, sometimes referred to CFF fonts or PostScript fonts;
2. outline fonts with quadratic Bézier curve, sometimes referred to TrueType fonts;
3. bitmap fonts.

Nowadays in digital typography an outline font is almost the only choice and no longer there is a relevant difference between a “PostScript” font and a “TrueType” font; there are some good commercial programs for creating and editing OpenType fonts for Windows and at least one GPL program, `fontforge`, which is known to run on Linux and Mac platforms. As an example, this is the `MALE AND FEMALE SIGN` Unicode character from font Symbola [1] with its points as they are shown by `fontforge`:



Symbola is an example of OpenType font with TrueType outlines which has been designed to match the style of Computer Modern font.

A brief note about bitmap fonts: among others, Adobe has published a “Glyph Bitmap Distribution Format (BDF)” [2] and with `fontforge` it’s easy to convert a bdf font into an opentype one *without* outlines. A fairly complete bdf font is http://unifoundry.com/unifont-5.1.20080820.bdf.gz: this file can be converted to an Opentype format `unifontmedium.otf` with `fontforge` and it can inspected with `showttf`, a `C` program from [3]. Here is an example of glyph U+26A5 MALE AND FEMALE SIGN:

```
Glyph 9887 (   uni26A5) starts at   492 length=17
height=12 width=8 sbX=4 sbY=10 advance=16
Bit aligned
        .....***
        ......**
        .....*.*
        ..***...
        .*...*..
        *.....*.
        *.....*.
        .*...*..
        ..***...
        ...*....
        ..***...
        ...*....
```

This font can also be viewed with `ftview` from freetype suite[4]:

```
#>ftview 16 unifontmedium.otf
```

and it can be embedded in a pdf document, but until today there isn’t still a pdf reader capable to display it. One can use Emacs to produce a PostScript file with bdf fonts embedded and then transform it into a pdf file, so that these bitmap fonts are managed as Type3 fonts, as shown in the example below:

```
(require 'ps-print)
   (require 'ps-mule)
   (setq ps-multibyte-buffer 'bdf-font)
```

CFF fonts come from Type1 fonts, where outlines are expressed in a subset of PostScript language in a dimensionless XY-space that is limited to the region with vertices (-16384,-16384) and (16383,16383), but usually it stays within a square measuring 1000 units.

With regard to this point in [7] we can read:

> “*The program inserts eight items (FontInfo, FontName, PaintType, FontType, FontMatrix, Encoding, FontBBox, and UniqueID) into the dictionary. The 1000 to 1 scaling in the FontMatrix as shown is typical of a Type 1 font program and is highly recommended.*”

and also in [6] at page 15 the default value for FontMatrix is `0.001 0 0 0.001 0 0`.
In [5] Adobe seems to enforce this position: at page 394 we can read (bold from the author):

> “*The glyph coordinate system is the space in which an individual character’s glyph is defined. All path coordinates and metrics are interpreted in glyph space.* ***For all font types except Type 3, the units of glyph space are one-thousandth of a unit of text space;*** *for a Type 3 font, the transformation from glyph space to text space is defined by a font matrix specified in an explicit FontMatrix entry in the font.*”

Starting from Adobe Reader 8, pdf documents that embed OpenType CFF fonts with upm different from 1000 units are shown in a wrong manner: for example this is the Italian word “assalire” with `IM_FELL_English_PRO_Roman` font with 2048 upm:

while the correct behaviour is

assalire

Note that the same font converted in a Type1 format *doesn’t show* this behaviour, so that the traditional way to manage fonts in ConTEXt-`mkii` works correctly with Adobe Reader 9 release and former (of course the correct encodings must be present in the system).

In the next section we will see how LuaTEX tackles this problem.

Inside LuaTEX

We will follow the evolution of `luatex` vers. `0.60.0` in a step-by-step fashion in a Linux box with the following file `test.tex` as input:

```
\pdfobjcompresslevel0
\pdfcompresslevel=0
\font\test=FeDPrm27C
\setuppagenumbering[location=]
\starttext
\test ab
\stoptext
```

We need a “nostrip” version of `luatex` which is easy to build from scratch with

```
#>build.sh --parallel  --nostrip
```

and then eventually regenerate the ConTEXt-`mkiv` format with

```
#>context --make --all
```

Note that when a new `luatex` is installed ConTEXt-`mkiv` is able to detect the new binary and hence rebuild the formats on the fly, but the author has found that sometimes erasing the internal cache can resolve some problems due to erroneous experimental configurations.

Next we will use the `ddd` debugger[8] with:

```
#>ddd --args
     "$ltxpath/luatex"
     --fmt="$ctxcache/cont-en"
     --lua="$ctxcache/cont-en.lui"
     --backend=pdf "$cwd/test.tex"
```

(one can see these values inspecting the first lines of `out`, where `out` is from `#> context test.tex  out`, while `$ltxpath` is the full path of luatex exec. and `$cwd` is the directory of `test.tex`)
A useful program is also `valgrind`[9] with its tool `callgrind`:

```
#>valgrind --tool=callgrind
      "$ltxpath/luatex"
       --fmt="$ctxcache/cont-en"
       --lua="$ctxcache/cont-en.lui"
       --backend=pdf "$cwd/test.tex"
```

and `kcachegrind`[10] to display the call graph.

Inside the BIG_SWITCH

The program `luatex` starts at

▸▸*source/texk/web2c/luatexdir/luatex.w*

```
424 int main(int ac, string * av)}
```

the so-called `/* The main program, etc.  */`.

Then there is an initialization function:

▸▸*source/texk/web2c/luatexdir/luatex.w*

```
435     lua_initialize(ac, av);
```

which takes care to manage `luatex` as (possibly) Lua interpreter only, parsing command line arguments, init the lua interpreter, setup synctex (“Synchronize TeXnology” cfr.[11]), and other things related to paths.

The heart of the program is `mainbody()`:

►►*source/texk/web2c/luatexdir/luatex.w*
```
    /* Call the real main program.  */
    mainbody();
```

Inside this function there are the initializations of various data structures and checks of hard-wired limits, the loading of the `cont-en.fmt` format and the `test.tex` input file, the initialization of the log file and, most important of all, the call of the main routine `main_control()`:

►►*source/texk/web2c/luatexdir/tex/mainbody.w*
```
  main_control();
```

which is in essence a big loop that gets a token at a time from input

►►*source/texk/web2c/luatexdir/tex/maincontrol.w*
```
  BIG_SWITCH:
   get_x_token();
```

and executes the appropriate function according to the `cur_cmd` value of the token:

►►*source/texk/web2c/luatexdir/tex/maincontrol.w*
```
    switch (abs(mode) + cur_cmd) {
    case hmode + letter_cmd:
    case hmode + other_char_cmd:
    case hmode + char_given_cmd:
    case hmode + char_num_cmd:
      if (abs(mode)+cur_cmd==hmode
                    +char_num_cmd){
            scan_char_num();
            cur_chr = cur_val;
:
:
    }/* end of the big |switch| statement */

    goto BIG_SWITCH;           /* restart */
}
```

There are 212 case-labels grouped into 80 different sets; it takes around fifty thousand calls to `get_x_token()` to go to the `do_final_end` of the program after `main_control()`:

►►*source/texk/web2c/luatexdir/tex/mainbody.w*
```
    flush_node(text_dir_ptr);
    final_cleanup(); /* prepare for death */
    close_files_and_terminate();
    FINAL_END:
    do_final_end();
}
```

because `cont-en` is a big format.

The TeX mission is to build boxes and put them in appropriate order: coming back to the `BIG_SWITCH` there are the most important cases:

►►*source/texk/web2c/luatexdir/tex/maincontrol.w*
```
/* Cases of |main_control| that build
    boxes and lists */
    case vmode + hrule_cmd:
    case hmode + vrule_cmd:
    case mmode + vrule_cmd:
/* The most important parts of |main_control|
    are concerned with \TeX's
    chief mission of box-making.
:
*/
```

but now the main focus is on `\font` primitive because with `\font\test=FeDPrm27C` we are defining a macro for font `FeDPrm27C.otf`:

►►*source/texk/web2c/luatexdir/tex/maincontrol.w*
```
/* Cases of |main_control| that
   do not depend on |mode| */
:
    case any_mode(def_font_cmd):
:
        prefixed_command();
```

which in turn calls `prefixed_command`:

►►*source/texk/web2c/luatexdir/tex/maincontrol.w*
```
void prefixed_command(void)
:
     case def_font_cmd:
/* Here is where the information
    for a new font gets loaded. */
         tex_def_font((small_number) a);
         break;
```

The macro `\test` is now defined and a new font structure is created for font file `FeDPrm27C`, but no box is made from this font until TeX reads the tokens `\test ab` from the input file. As seen before, the most important part of TeX is making boxes, so `\test ab` will end in a box inside a page, which in turn is another `\vbox` emitted by the output routine. ConTeXt-mkiv has a complex page layout, so the box emitted is also complex: in this case there are 244 items (called "nodes") in this box and near the end there are the "a" and "b" characters in `\test` font.

```
1\vbox(772.77686+0.0)x426.78743, direction TLT
2.\glue -72.26999
3.\hbox(845.04684+0.0)x426.78743, direction TLT
4..\whatsit
5...\localinterlinepenalty=0
6...\localbrokenpenalty=0
7...\localleftbox=null
8...\localrightbox=null
9..\hbox(0.0+0.0)x0.0, direction TLT
10..\glue -72.26999
:
:
```

```
215.....................\pdfliteral page0 g 0 G
216.....................\test a
217.....................\test b
218.....................\penalty 10000
243..\glue(\parfillskip) 0.0 plus 1.0fil
244..\glue(\leftskip) 0.0
```

The function `box_end` manages all types of boxes that make up the page and also the final vbox of the page (cfr. nr. 1 of the list above) by calling the `ship_out` function:

►► *source/texk/web2c/luatexdir/tex/maincontrol.w*

```
1282 @ The global variable |cur_box| will point
     to a newly-made box. If the box
1283 is void, we will have |cur_box=null|.
     Otherwise we will have
1284 |type(cur_box)=hlist_node| or |vlist_node|
     or |rule_node|; the |rule_node|
1285 case can occur only with leaders.
1286
1287 @c
1288 halfword cur_box; /* box to be placed
                         into its context */
1289
1290
1291 @ The |box_end| procedure does
     the right thing with |cur_box|, if
1292 |box_context| represents
     the context as explained above.
1293
1294 @c
1295 void box_end(int box_context)
:
1364        } else
1365         ship_out(static_pdf, cur_box, true);
1366     }
1367 }
```

The `ship_out` function manages two types of nodes, the vertical one and the horizontal one:

►► *source/texk/web2c/luatexdir/pdf/pdfshipout.w*

```
46 @ |ship_out| is used to shipout a box to PDF
   or DVI mode.
47 If |shipping_page| is not set then the
   output will be a Form object
48 (only PDF), otherwise it will be a Page object.
49
50 @c
51 void ship_out(PDF pdf, halfword p,
              boolean shipping_page)
:
273 switch (type(p)) {
274  case vlist_node:
275      vlist_out(pdf, p);
276      break;
277  case hlist_node:
278      hlist_out(pdf, p);
279      break;
280  default:
281      assert(0);
282 }
```

In this case it's a horizontal node and the program sends one char to the output back-end:

►► *source/texk/web2c/luatexdir/pdf/pdflistout.w*

```
313 void hlist_out(PDF pdf, halfword this_box)
:
382                output_one_char(pdf, font(p),
                              character(p));
```

For ConTEXt-`mkiv` the pdf back-end is the default output back-end, but one can choose the dvi back-end as well:

►► *source/texk/web2c/luatexdir/pdf/pdffont.w*

```
44 @ The following code typesets
   a character to PDF output
45
46 @c
47 void output_one_char(PDF pdf,
              internal_font_number ffi, int c)
:
70         backend_out[glyph_node] (pdf, ffi, c);
           /* |pdf_place_glyph(pdf, ffi, c);| */
```

Given that it's the first time that the font is used, a `setup_fontparameters` is needed:

►► *source/texk/web2c/luatexdir/pdf/pdfglyph.w*

```
173 void pdf_place_glyph(PDF pdf,
              internal_font_number f, int c)
:
182         if (f != pdf->f_cur)
183             setup_fontparameters(pdf, f);
```

This is the first part (of two) where `luatex` manages non standard fontmatrix: at line 68 with

`u = (float) (font_units_per_em(f)/ 1000.0);`

the `fonts_units_per_em` matrix is "normalized" in a way that, in essence, the font appears to be loaded at $\mathtt{font\_design\_size} \times \frac{1000}{2048}$ i.e. $10 \times \frac{1000}{2048} = 4.8828\mathrm{bp}$. It's important to note that there aren't other "re-scaling" actions, (no "outline re-scaling" for example) so that rounding errors are limited.

►► *source/texk/web2c/luatexdir/pdf/pdfglyph.w*

```
59 static void setup_fontparameters(PDF pdf,
                  internal_font_number f)
60 {
61     float slant, extend, expand;
62     float u = 1.0;
63     pdfstructure *p = pdf->pstruct;
64     /* fix mantis bug \#
          0000200 (acroread "feature") */
```

```
if ((font_format(f) == opentype_format ||
     (font_format(f) == type1_format &&
            font_encodingbytes(f) == 2))
    && font_units_per_em(f) > 0)
    u = (float) (font_units_per_em(f)
                 / 1000.0);
pdf->f_cur = f;
p->f_pdf = pdf_set_font(pdf, f);
p->fs.m = lround((float) font_size(f) / u
        / one_bp * ten_pow[p->fs.e]);
slant = (float) font_slant(f)
        / (float) 1000.0;
extend = (float) font_extend(f)
        / (float) 1000.0;
expand = (float) 1.0
        + (float) font_expand_ratio(f)
        / (float) 1000.0;
p->tj_delta.e = p->cw.e - 1;
/* "- 1" makes less
   corrections inside []TJ */
/* no need to be more precise
   than TeX (1sp) */
while (p->tj_delta.e > 0
       && (double) font_size(f)
    / ten_pow[p->tj_delta.e + e_tj] < 0.5)
    p->tj_delta.e--;        /* happens for
                     very tiny fonts */
assert(p->cw.e >= p->tj_delta.e);
/* else we would need, e. g., |ten_pow[-1]| */
p->tm[0].m =
  lround(expand * extend
        * (float) ten_pow[p->tm[0].e]);
p->tm[2].m = lround(slant
        * (float) ten_pow[p->tm[2].e]);
p->k2 =
    ten_pow[e_tj +
            p->cw.e]
            / (ten_pow[p->pdf.h.e]
            * pdf2double(p->fs) *
              pdf2double(p->tm[0]));
}
```

With this step only and no other correction, xpdf and mupdf readers will show the same wrong picture seen before for Adobe Reader because there is an effective mismatch in font dimensions: the font matrix does not match the effective dimensions of each glyph in the text. Note that until here there are no glyphs on the output, only nodes, because this is a back-end issue. For example, at the end of this part the text "ab" (glyphs number 0x0044 and 0x0045) is placed into the output pdf:

▸▸*test.pdf*

```
14 0 obj <<
/Length 86
>>
stream
0 g 0 G
0 g 0 G
BT
/F44 4.86457 Tf 1 0 0 1 70.867 702.3845
  Tm [<0044>-430<0045>]TJ ET
```

In the next subsection it's described the second and last part that corrects this behaviour.

Outside the BIG_SWITCH

The second part gets into play when there are no more boxes to manage, i.e. when `\stoptext` macro is executed. The program `luatex` is now after `main_control`:

▸▸*source/texk/web2c/luatexdir/tex/mainbody.w*

```
    main_control();
    flush_node(text_dir_ptr);
    final_cleanup(); /* prepare for death */
    close_files_and_terminate();
    FINAL_END:
    do_final_end();
}
```

`flush_node` makes sure that all remaining nodes, if any, are deleted from memory, while the `final_cleanup` function is called when `luatex` has expanded `\stoptext` which in turn calls `\end` (it's also called when dumping formats):

▸▸*source/texk/web2c/luatexdir/tex/mainbody.w*

```
@ We get to the |final_cleanup| routine
  when \.{\\end} or \.{\\dump} has
been scanned and |its_all_over|\kern-2pt.

@c
void final_cleanup(void)
:
```

The function `close_files_and_terminate` is the most important because it translates TEX data structures to back-end data structures (pdf in this case). It's hard here to keep track of every step but in essence a pdf file is a collection of objects organized in tree-like data structures (a tree plus attributes); all objects are indexed by an `xref` table and they can be referenced to from other objects: for example these are the objects that make reference to some page:

▸▸*test.pdf*

```
:
16 0 obj <<
/Type /Pages
/Count 1
/Kids [13 0 R]
>> endobj
22 0 obj <<
```

```
q>> endobj
23 0 obj <<
/Type /Catalog
/Pages 16 0 R
/Names 22 0 R
/Version /1.6 /PageMode /UseNone /Metadata 11 0 R
>> endobj
```

(a quick but good overview of pdf is [12]).

Given that currently `luatex` output mode is pdf (`OMODE_PDF`) then the `close_files_and_terminate` function calls `finish_pdf_file`:

▸▸*source/texk/web2c/luatexdir/tex/mainbody.w*

```
501 void close_files_and_terminate(void)
:
547     switch (pdf->o_mode) {
548     case OMODE_NONE: /* during initex run */
549         break;
550     case OMODE_PDF:
551         if (history == fatal_error_stop) {
552             remove_pdffile(pdf);
553             print_err
554                 (" ==> Fatal error occurred,
                  no output PDF file produced!");
555         } else
556             finish_pdf_file(pdf,
                            luatex_version,
                            get_luatexrevision());
557         break;
558     case OMODE_DVI:
```

In the `finish_pdf_file` function there is the code that manages the font, `do_pdf_font` (line 2246):

▸▸*source/texk/web2c/luatexdir/pdf/pdfgen.w*

```
2182 @ Now the finish of PDF output file.
     At this moment all Page objects
2183 are already written completely to
     PDF output file.
2184
2185 @c
2186 void finish_pdf_file(PDF pdf,
               int luatex_version,
       str_number luatex_revision)
2187 {
:
2241          k = pdf->head_tab[obj_type_font];
2242          while (k != 0) {
2243              f = obj_info(pdf, k);
2244              assert(pdf_font_num(f) > 0);
2245              assert(pdf_font_num(f) == k);
2246              do_pdf_font(pdf, f);
2247              k = obj_link(pdf, k);
2248          }
2249          write_fontstuff(pdf);
```

Font `FeDPrm27C.otf` is an OpenType font, hence it can manage up to 65536 glyphs that need 2 bytes to be indexed, (`font_encodingbytes(f) == 2`); the font dictionary is created at line 806, after some setups:

▸▸*source/texk/web2c/luatexdir/font/writefont.w*

```
730 void do_pdf_font(PDF pdf,
    internal_font_number f)
731 {
:
741     if (font_encodingbytes(f) == 2) {
742         /* Create a virtual font map entry,
             as this is needed by the
743          rest of the font inclusion mechanism.
744          */
:
805         set_cidkeyed(fm);
806         create_cid_fontdictionary(pdf, f);
807
808         if (del_file)
809             unlink(fm->ff_name);
810
811     }
```

With function `create_cid_fontdictionary` `luatex` starts to write to the pdf file the current font parameters (there is only one in `test.tex`) , i.e. the chars widths, the font description and the font dictionary (which contains informations like font type, subtype etc.):

▸▸*source/texk/web2c/luatexdir/font/writefont.w*

```
937 static void create_cid_fontdictionary(PDF pdf,
                          internal_font_number f)
938 {
:
956     write_cid_charwidth_array(pdf, fo);
957     write_fontdescriptor(pdf, fo->fd);
958
959     write_cid_fontdictionary(pdf, fo, f);
```

The function `write_fontdescriptor` manages the embedding of the font file that contains the actual glyphs:

▸▸*source/texk/web2c/luatexdir/font/writefont.w*

```
471 static void write_fontdescriptor(PDF pdf,
                              fd_entry * fd)
472 {
:
498     if (is_fontfile(fd->fm)
         && is_included(fd->fm))
499         write_fontfile(pdf, fd);  /* this will
           set |fd->ff_found| if font
           file is found */
```

At this level there is not so much difference between a Type1 font, a TrueType font or their counterparts in OpenType, so `write_fontfile` manages all of them: in this case `FeDPrm27C.otf` is an OpenType “Type 1”-like font, and is managed by `writetype0`:

▸▸*source/texk/web2c/luatexdir/font/writefont.w*

```
static void write_fontfile(PDF pdf, fd_entry * fd)
{
    assert(is_included(fd->fm));
    if (is_cidkeyed(fd->fm)) {
        if (is_opentype(fd->fm))
            writetype0(pdf, fd);
```

`writetype0` opens the file to read font parameters, i.e. the table “head” (Font header), “hhea” (Horizontal header), “PCLT” (PCL 5 data), “post” (PostScript information) and then reads the glyphs with `read_cff` (the table “CFF”, PostScript font program (compact font format)) to put them into the pdf file:

▸▸*source/texk/web2c/luatexdir/font/writetype0.w*

```
void writetype0(PDF pdf, fd_entry * fd)
{
:
    /* copy font file */
    tab = ttf_seek_tab("CFF ", 0);

    /* TODO the next 0 is a subfont index */

    cff = read_cff(tt_buffer + ttf_curbyte,
                  (long) tab->length, 0);
```

Before the glyphs are actuelly put into pdf, `luatex` needs to read, among others, the font dictionary `DICT`(cfr. [6]):

▸▸*source/texk/web2c/luatexdir/font/writecff.w*

```
cff_font *read_cff(unsigned char *buf,
                   long buflength, int n)
{
:
    cff->topdict = cff_dict_unpack(idx->data
                       + idx->offset[n] - 1,
           x->data + idx->offset[n + 1] - 1);
```

and it’s just here that, with `add_dict`

▸▸*source/texk/web2c/luatexdir/font/writecff.w*

```
cff_dict *cff_dict_unpack(card8 * data,
                          card8 * endptr)
{
    cff_dict *dict;
    int status = CFF_PARSE_OK;

    stack_top = 0;

    dict = cff_new_dict();
    while (data < endptr && status
           == CFF_PARSE_OK) {
        if (*data < 22) {       /* operator */
       add_dict(dict, &data, endptr, &status);
:
```

the FontMatrix is reset to 1000upm:

▸▸*source/texk/web2c/luatexdir/font/writecff.w*

```
static void add_dict(cff_dict * dict,
                       card8 ** data,
         card8 * endptr, int *status)
{
    int id, argtype, t;

    id = **data;
    if (id == 0x0c) {
:
  if (t > 3 && strcmp(dict_operator[id].opname,
     "FontMatrix") == 0) {
  /* reset FontMatrix to [0.001 * * 0.001 *],
     fix mantis bug \# 0000200
     (acroread "feature") */
(dict->entries)[dict->count].values[0] = 0.001;
(dict->entries)[dict->count].values[3] = 0.001;
            }
            dict->count += 1;
```

It’s important to note that not all of these operations must be repeated — for the first used glyph.

Conclusion

LuaTEX with ConTEXt-`mkiv` is the first TEX system that manages opentype CFF fonts with unusual upm without transforming them in an equivalent Type1, hence avoiding the need of an explicit encoding map. By examining the luatex program internals to see how this is implemented, a number of small changes are shown that minimize the necessary recalculations so as to keep rounding errors to a minimum. Anyway this solution does not preclude an automatic conversion from OpenType CFF to Type1 format in the future, which is possible but more complicated.

References

All links were verified between 2010.04.02 and 2010.04.09.

[1] http://users.teilar.gr/~g1951d/Symbola253.zip
[2] http://www.adobe.com/devnet/font/pdfs/5005.BDF_Spec.pdf
[3] http://fontforge.cvs.sourceforge.net/viewvc/fontforge/fontforge/fonttools/
[4] http://www.freetype.org/
[5] http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf
[6] http://www.adobe.com/devnet/font/pdfs/5176.CFF.pdf
[7] http://www.adobe.com/devnet/font/pdfs/T1_SPEC.PDF
[8] http://www.gnu.org/software/ddd
[9] http://valgrind.org/
[10] http://kcachegrind.sourceforge.net/cgi-bin/show.cgi
[11] http://itexmac.sourceforge.net/SyncTeX.html
[12] http://wiki.contextgarden.net/images/a/a7/Eurotex06he.pdf

Luigi Scarso

Nieuws van CTAN

Een uittreksel uit de recente bijdragen in het CTAN archief

Abstract
Dit artikel beschrijft een aantal recente bijdragen uit het CTAN archief (en andere bronnen op het Internet). De selectie is gebaseerd op wat ik zelf interessant vind en wat ik denk dat voor veel anderen interessant is. Het is dus een persoonlijke keuze. Het heeft niet de bedoeling om een volledig overzicht te geven.

Keywords
TEX, LATEX, packages, CTAN, classes, graphics, pstricks, programma's.

Inleiding, en een persoonlijke noot

Het was oorspronkelijk mijn bedoeling om in iedere aflevering van MAPS deze bijdrage (Nieuws van CTAN) te leveren, maar doordat we nogal druk zijn geweest met een aantal veranderingen in de persoonlijke sfeer (nieuw huis bouwen, verhuizen, pensioen, en nu een jaar naar Zuid Amerika) kwam er telkens niet van. Ik probeer het nu weer op te pakken en dit artikel ben ik begonnen te schrijven op de boot van Buenos Aires naar Colonia (Uruguay). En ik ben verder gegaan bij de watervallen van Iguazu, op de grens van Argentinië, Brazilië en Paraguay en in de bus in Paraguay. Tenslotte is het artikel afgemaakt in Santa Cruz, Bolivia. Bij het schrijven was ik wel enigszins onthand door gebrekkige of ontbrekende internetverbindingen.

Ik vind dat ik ook niet meer rekening hoef te houden met de vorige afleveringen en ik doe dan ook alsof deze er niet zijn geweest.

Pstricks

Als je de CTAN aankondigingen van de laatste paar maanden doorleest dan valt op dat er zeer veel bijdragen zijn van het oude getrouwe `pstricks`. Misschien even ter opfrissing: `pstricks` is een verzameling macro's om op Postscript gebaseerde tekeningen op te nemen in een TEX file. In principe is het geschikt voor iedere vorm van TEX, dus ook LATEX of context. Er is ook een bijbehorend LATEX pakket om het gebruik in LATEX documenten gemakkelijker te maken.

Daarbij worden de tekeningen niet buiten TEX klaargemaakt en dan ingevoegd, zoals het geval zou zijn als je een tekenprogramma gebruikt, maar de Postscript code wordt door middel van TEX commando's gegenereerd. Dit heeft het voordeel dat de gegenereerde Postscript af kan hangen van variabelen in het document. Zo kun je bijvoorbeeld pijltjes trekken tussen twee punten in het document die bepaald worden door de lopende tekst.

Nu vermoed ik dat het aantal mensen dat met Postscript werkt de laatste jaren drastisch in afgenomen. In plaats daarvan is er een verschuiving geweest naar PDF. Er is echter een belangrijk verschil tussen Postscript en PDF: hoewel beide bedoeld zijn om documenten (of pagina's) te beschrijven, is Postscript behalve een pagina-opmaaktaal ook een programmeertaal en PDF is dat niet. Je kunt daarom allerlei berekeningen in Postscript laten doen die in PDF niet mogelijk zijn. Als je PDF gebruikt moeten de berekeningen dus in TEX gedaan worden maar helaas zijn dan soms niet alle gegevens voorhanden, bijvoorbeeld de positie van een bepaalde tekst op de pagina. Bovendien is het in TEX moeilijk om berekeningen met voldoende precisie te doen en zijn verschillende handige wiskundige operaties niet voorhanden. Er zijn ook wel weer manieren om hieromheen te werken maar dat maakt het soms wel omslachtiger. Overigens biedt luaTEX hier wel weer nieuwe mogelijkheden.

In het licht van bovenstaande is het aan de ene kant verbazingwekkend dat er kennelijk nog zo hard aan `pstricks` gewerkt wordt, aan de andere kant vanwege die extra mogelijkheden van Postscript kan het ook weer aantrekkelijk zijn om het te gebruiken.

Bij het gebruik van op PDF gebaseerde versies van TEX (zoals PDFLATEX) is het probleem echter dat daar geen Postscript specials in gebruikt kunnen worden. Hiervoor zijn echter een aantal oplossingen bedacht die er op neer komen dat eerst een aantal keren TEX met DVI-output gebruikt wordt, en de output hiervan omgezet wordt naar PDF. De resulterende PDF figuren worden dan in een laatste run ingevoegd in het uiteindelijke document. Een van deze oplossingen is een Perl-script `pst2pdf`, die van een LATEX-document de `pspicture` omgevingen eruit vist, die door gewoon LATEX heen haalt en de resulterende DVI-output met `dvips` en `ghostscript` omzet naar PDF-bestanden. Dit script heb ik ook gebruikt om de figuren in dit artikel voor te bereiden. (`Pspicture` is vergelijkbaar met de standaard `picture` omgeving van LaTeX, maar dan voor de `pstricks` tekeningen.)

`Pstricks` was in eerste instantie ontwikkeld door Timothy Van Zandt, terwijl later het onderhoud overgenomen is door Denis Girou, Sebastian Rahtz, Rolf Niepraschk en Herbert Voss, waarbij momenteel voornamelijk de laatste ermee bezig is.

`Pstricks` is opgedeeld in een groot aantal deelpakketten zodat je kunt kiezen wat je nodig hebt. Een aantal van deze pakketten is door anderen geschreven.

Pstricks.tex en pstricks.pro

Dit zijn de basispaketten, terwijl de andere uitbreidingen hiervan zijn. `Pstricks` is nu op versie 2.04 aangeland. Versie 2 is een significante vernieuwing, en het feit dat de versie nu op 2.04 staat geeft aan dat er intussen al weer kleine veranderingen zijn aangebracht voornamelijk in de sfeer van oplossingen voor bugs.

Pst-add

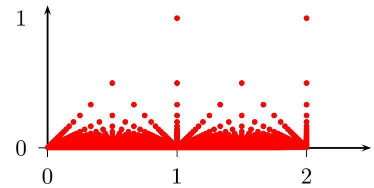
`Pst-add` is een pakket dat een aantal elementaire uitbreidingen op `pstricks` levert zoals pijlen en speciale curves. Sommige van deze uitbreidingen zijn of waren experimenteel. Met de nieuwe versie van `pstricks` zijn veel van deze experimentele uitbreidingen verhuisd naar standaard pakketten zoals `pstricks` zelf of `pst-node` en `pst-plot`.

Pst-func

`Pst-func` bevat de code om een groot aantal wiskundige functies te plotten. Dit is vooral van betekenis voor mensen in de exacte vakken, met name wiskundigen en natuurkundigen, maar er zitten ook functies bij die voor economen van belang zijn. Verder ook macro's voor het tekenen van Bezier-curven die nuttig zijn voor het tekenen van kromme lijnen in tekeningen, dus voor grafisch ontwerp. Ook zit er een groot aantal functies uit de statistiek bij, waardoor dit ook voor andere wetenschappers van belang kan zijn.

Als voorbeeld de Thomae functie, ook wel popcorn functie genoemd.

```
\usepackage{pst-func}

\psset{unit=2cm}
\begin{pspicture}(-0.1,-0.2)(2,1.15)
    \psaxes{->}(0,0)(2.5,1.1)
    \psThomae[dotsize=2.5pt,linecolor=red]%
             (0,2){300}
\end{pspicture}
```



Pst-sigsys

Dit pakket bevat macro's die voor mensen die zich bezig houden met signaalverwerking nuttig zijn. Verschillende diagrammen uit deze discipline kunnen hiermee gegenereerd worden.

Pst-circ

`Pst-circ` kan gebruikt worden om elektronische schakelingen te tekenen.

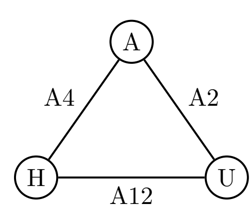
Pst-node

`Pst-node` is een zeer fundamenteel sub-pakket van `Pstricks` waarmee nodes en verbindingen tussen nodes getekend kunnen worden. Hiermee kunnen netwerkdiagrammen getekend worden. Dit soort diagrammen wordt in heel veel disciplines gebruikt. Zo'n diagram bestaat uit een aantal nodes (punten, cirkels of vierkantjes bijvoorbeeld) waartussen verbindingen aanwezig kunnen zijn. Een aantal voorbeelden hiervan:

- Steden als nodes met de verbindingswegen als verbindingen tussen de nodes
- Taken in een project met de afhankelijkheden tussen deze taken als verbindingen
- Personen in een organisatie met de relaties tussen deze personen als verbindingen.

Het aantal toepassingen van dit soort structuren is onbegrensd.

```
\begin{postscript}
  \psmatrix[mnode=circle,colsep=1]
  & A \\
  H & & U
  \endpsmatrix
  \psset{shortput=nab,labelsep=3pt}
  \ncline{1,2}{2,3}^{A2}
  \ncline{2,1}{2,3}_{A12}
  \ncline{2,1}{1,2}^{A4}
\end{postscript}
```



In dit voorbeeld is geen `pspicture` omgeving gebruikt maar een matrix waarin de nodes gezet worden met `\psmatrix`. Naar de nodes wordt dan later verwezen met de coördinaten in deze matrix: (1,2) en dergelijke. Ik heb hier voor de verandering de gewone TEX-commando's `\psmatrix` en `\endpsmatrix` gebruikt; dit kan echter ook met de `psmatrix` omgeving in LATEX. Verder heb ik er de `postscript`-omgeving omheen gezet. Dit is voor `pst2pdf` omdat er geen `pspicture`-omgeving gebruikt is. Zo kan `pst2pdf` uitvinden welk stuk `pstricks` code is.

Pst-coil

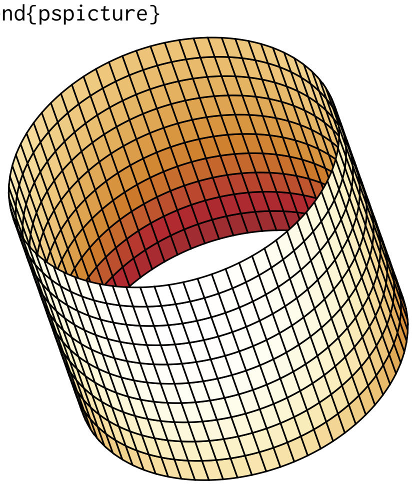
`Pst-coil` wordt gebruikt om zigzaglijnen, spoelen en dergelijke te tekenen. Het tekenen van spoelen is nuttig in toepassingen van elektronica, maar zigzaglijnen zijn in allerlei toepassingen nuttig te gebruiken. Dit pakket kan ook sinus-vormige verbindingslijnen tussen nodes te trekken in een diagram.

Pst-3d en pst-3dplot

`Pst-3dplot` geeft macro's om 3-dimensionale objecten te tekenen. Dit houdt in dat ook andere objecten zoals teksten en curves schuin getekend kunnen worden omdat je er in 3-d schuin tegenaan kijkt. Denk bijvoorbeeld aan een kubus in 3-d waar aan de zijkant (die je schuin ziet) een tekst staat. Postscript als grafische taal heeft al deze transformaties ingebouwd. Overigens heeft PDF dit ook. Al het echte rekenwerk voor deze operaties wordt dus door Postscript uitgevoerd. `Pst-3d` bevat een aantal macro's die door `pst-3dplot` gebruikt worden.

Hierbij een voorbeeld van een 3d-cilinder:

```
\usepackage{pst-3dplot}

\begin{pspicture}(-3,-2)(3,6)
\psset{Beta=60}
\psCylinder[RotX=10,increment=5]{3}{5}
\end{pspicture}
```



Pst-fractal

Met `pst-fractal` kunnen fractale figuren getekend worden. Bekende fractals zijn de Mandelbrot-figuren en sneeuw-vlokachtige krommen. Een fractal is kort gezegd een figuur die bestaat uit onderdelen die zelf weer verkleinde versies zijn van de hele figuur. In de wiskundige betekenis gaat dit tot het oneindige door. In een praktische toepassing houdt het natuurlijk na een aantal iteraties op. In de levende natuur komen ook veel fractalachtige objecten voor. Als voorbeeld de zogenaamde Fern-fractal, die varen-achtige structuren geeft.

```
\usepackage{pst-fractal}

\begin{pspicture}(-3,0)(3,11)
  \psFern[scale=10,maxIter=10000]
\end{pspicture}
```

Pst-jtree

`Pst-jtree` is een pakket voor het tekenen van boomstructuren die vooral nuttig zijn voor linguisten, bijvoorbeeld ontleedbomen. Overigens maken ook informatici en anderen uitgebreid gebruik van boomstructuren.

Pst-gantt

`Pst-gantt` is een pakket voor het tekenen van Gantt diagrammen die gebruikt worden in projectmanagement.

Dit was slechts een gedeelte van de `pstricks`-pakket. Bovenstaande pakketten zijn recent vernieuwd maar er zijn er nog veel meer die oudere versies hebben.

De pakketten zijn alle te vinden onder (subdirectories van) `CTAN:/graphics/pstricks/`.

Eqparbox

Soms is het nodig dat een aantal blokken tekst dezelfde breedte krijgen, ongeacht waar ze staan in het document. Het eqparbox-pakket definieert commando's \eqparbox, \eqmakebox, \eqframebox en \eqsavebox, die net zoals resp. \parbox, \makebox, \framebox en \savebox werken, maar in plaats van een breedte, een label als parameter krijgen. Alle boxen met hetzelfde label worden dan opgerekt tot de grootste breedte in de serie.

CTAN:/macros/latex/contrib/eqparbox

Isodate

Het isodate-pakket maakt het mogelijk datums uit te voeren in tien verschillende uitvoer-formaten, sommige taal-afhankelijk.

CTAN:/macros/latex/contrib/isodate/

marginnote

Het pakket marginnote geeft een commando \marginnote, dat werkt als \marginpar maar het kan op meerdere plaatsen gebruikt worden. Bijvoorbeeld binnen een float, een voetnoot of een frame dat met het framed-pakket gemaakt wordt. In deze omgevingen hebben \marginpars de neiging om gewoon in een zwart gat te verdwijnen.

CTAN:/macros/latex/contrib/marginnote

Pdfrack

Een oude getrouwe onder mensen die met grafieken en dergelijke werken is psfrag. Dit pakket maakt het mogelijk om in EPS-figuren de teksten te vervangen door complete stukken TEX. Grafieken kunnen gemaakt worden met een groot aantal wetenschappelijke pakketten en bijvoorbeeld Excel, maar deze kunnen meestal geen TEX-teksten gebruiken. Met psfrag kun je dan simpele teksten in de grafieken gebruiken, zoals A, B, C, en met psfrag kunnen deze worden vervangen door echte stukken TEX. Alleen, net zoals bij pstricks, kan dit niet met PDFLATEX gebruikt worden. Het pakket pdfrack geeft je nu de mogelijkheid om dit toch te doen. De auteur noemt het een 'hack'.

CTAN:/support/pdfrack

Xypdf

XY-pic is ook een pakket dat in de loop der jaren zijn diensten bewezen heeft. Het is een zeer uitgebreid pakket voor het maken van diagrammen met nodes en pijlen ertussen (zoiets als bij pst-node hierboven beschreven is maar dan zeer uitgebreid). Het pakket is op zich niet direct afhankelijk van Postscript, maar om goede kwaliteit lijnen, curven, cirkels en pijlen te krijgen gebruikt het wel de mogelijkheden van Postscript. Voor het gebruik met PDF moest dan teruggevallen worden op een methode met een mindere kwaliteit, zoals het opbouwen van deze elementen met behulp van puntjes en dergelijke. Het pakket xypdf verbetert dit nu, omdat hiervoor dan echte PDF-elementen gebruikt worden. Het werkt zowel met PDFLATEX als met LATEX gevolgd door dvipdfm of dvipdfmx.

CTAN:/macros/latex/contrib/xypdf

Programma's

Tenslotte wil ik nog even twee leuke dingen noemen die niet op CTAN voorkomen maar handige programmaatjes (apps) zijn voor de iPhone of iPod Touch. Je kunt ze vinden in de iTunes store.

Detexify

Detexify is een verbazingwekkend programmaatje waarin je op het scherm een LATEX-symbool kunt tekenen waarna het een aantal LATEX-commando's geeft die dit symbool produceren of iets dat er gelijkenis mee vertoont. Het heeft wel een internetverbinding nodig om de match te vinden. Dit programma is dan ook afgeleid van een applicatie op het internet waarmee je hetzelfde kunt doen op je computer. Probeer het eens uit: http://detexify.kirelabs.org/classify.html

LaTeX Help

Deze iPhone/iPod Touch app geeft je een groot aantal LATEX-commando's en -symbolen, kant en klaar met voorbeelden. Niet alleen de standaard LATEX-commando's maar ook die van het amsmath-pakket. Het kon nog wat uitgebreider: uitleg over de standaard LATEX omgevingen ontbreekt bijvoorbeeld.

Piet van Oostrum
http://www.pietvanoostrum.com
piet@vanoostrum.org