Bijlage 5

Tiling in PostScript and METAFOONT - Escher’s wink

Kees van der Laan

abstract
Drawing tilings by computer is discussed. Examples are borrowed from literature. New are their included METAFOONT and PostScript programs, with sometimes a new variation of a picture.

keywords
Apollonius, Computer art, Douat, Dürer, education, EPS, Escher, graphics, METAFOONT/Post, outline, Kepler, Mondrian, parquet, Penrose, PostScript, puzzles, Soddy, tiling, Truchet, turtle graphics, zero finding.

1 Introduction
Children like to play, like jigsaw puzzles. Tiling avant la lettre, where the form and color of the pieces matter.

Example (Part of Escher’s Sun and Moon)

Dark birds in bright daylight or white birds in the dark? What has this got to do with tiling? Well ... it is a tiling of free forms. When considered as a non-tight tiling of either dark or white birds the others are spurious. Intriguing. Underneath is a grid of triangles, reshaped into birds.

Escher anthropomorphised tiling by introducing creatures as decoration of the tiles as opposed to traditional tiling. Tiles used to take strict geometrical patterns especially the Islamic ones because of religion.

Escher considered lines as to be 2-sided, and exploited this in depth. Seen from one side the line is the boundary of a figure and looked upon from the other side gives meaning to another figure.

Tiling has all to do with filling up the plane. It is an old human activity of which beautiful results have been preserved, for example in the Alhambra. In this note a few examples of tilings are shown next to their codes — METAFOONT, or POSTSCRIPT—for drawing them by computer.

Intriguing and captivating are Escher’s contributions. In some patterns he not only translates but also rotates and reflects. In others, the so-called limit cycles, he shrinks the size of the basic element gradually, suggestively infinitely. In his metamorphoses tiles change their form in crescendo.

Example (Classification of homogeneous tiles)
The following homogeneous, Archimedean tiles are classified in mathematical tiling theory by the polygons at each vertex. The left is denoted by $[4, 8, 8]$ — a square and 2 octagons join at each corner — and the right one by $[3, 4, 6, 4]$.2

However, no attempt will be made to treat mathematical tiling theory. It seems that a constructive computer-oriented approach and programming terminology are needed. I found it advantageous to concentrate on non-tight tilings with spurious elements. For example the right figure — extended infinitely — can be drawn by just squares appropriately positioned. The result will contain the spuri-

1 A thorough reference about tiling is: Tilings and patterns by Branko Grünbaum & G C Shephard. Freeman, New York. ISBN 0-7167-1193-1. Martin Gardner in his Scientific American notes has popularized various (mathematical) puzzles. Bouwkamp, a Dutch physicist, has worked on 3D tilings. MacGillavry, a Dutch crystallographer, has analyzed Escher’s use of symmetries and correlated this to crystal symmetries. Coxeter, a Canadean mathematician, has worked with Escher and contributed to mathematical tiling theory, if not for the limit cycles.

2 Do you recognize a variant of the Pythagorean tree in the right one? Isn’t it amazing? Pythagorean trees will be treated in the note on fractals.
ous triangles and hexagons. To draw Pythagorean trees needed a lifetime in the 40-ies. I don’t know how much time it took to draw tilings, but with the computer it is just a matter of programming, which is substantial less especially when we can build from templates.

1.1 Coding

The coding of tilings comes down to finding a basic element—a tile—and to make compositions of rotated and/or translated copies. Because these symmetry operations can be easily programmed for a computer, drawing and designing tilings can be seen as computer art nowadays.

In contrast with general POSTSCRIPT codes which are usually generated by programs just for use, my codes are concise, consistent, educational, procedural, and next best to literate.

The POSTSCRIPT codes are ready for use. The METAFONT codes don’t build a character, just the inherent aspects are shown, to illustrate differences with coding in POSTSCRIPT. For use adaptations are needed reflecting your computing environment, be it to build a character from the code and export this etc., or to make a MetaPost code from it, and so on.

Not all codes have been included, especially the more elaborate ones, mostly of composite tilings, have been omitted. Nor are there METAFONT codes for all pictures. The reason is that I started with METAFONT but experienced later on POSTSCRIPT, and when the note grew only POSTSCRIPT codes were developed, which for these problems is sufficient and practical enough, given my situation. It is hoped that the included METAFONT codes serve their purpose.

Fractal tilings and space filling curves—well... mathematical curves—will be treated in separate notes.

All the enclosed pictures have POSTSCRIPT codes; none has been scanned.

Conventions I’ve adopted the following conventions in coding POSTSCRIPT and hope by stating them it will ease the reading of the codes.

1. Constants are at the beginning and have short names. The prefix m (minus) denotes the negative value, h denotes half the value. The postfix loc denotes a quantity to be used locally, especially in recursion. So mhslloc denotes −.5s, to be used locally.

2. iname def, lays hands on the value on the top of the stack. This is used to reuse an argument (by name), supplied on the stack (before the invoke of an operator).

Generally, intelligibility is served by appropriate spacing, and when spacing is absent either it is trivial or spacing is not relevant for understanding.

The following selection of structuring elements are reminders for those not familiar with POSTSCRIPT.

1. definitions
   /iname... def
   /iname{...} def

2. structures
   gsave...grestore, the graphics state delimiters
   boolean {truepart} if
   boolean {truepart}{falsepart} ifelse
   value {loopbody} repeat
   begin step end {loopbody} for
   array {loopbody} forall
   {loopfirstpart boolean} if
   {loopsecondpart} loop

Example (Tangram)

Another way of looking might emerge from playing with Tangram-like puzzles, to combine the (holy) 7 pieces into free forms, stimulating phantasy.

In POSTSCRIPT the Tangram tablet can be drawn as follows.

3. I did not pluck from the net codes programmed by various people, with as a consequence that the programs enjoy a common approach and programming style facilitating intelligibility.

4. My METAFONT codes are essentially compatible with MetaPost codes, because I pay attention to use only the common features. For example, I don’t rely on the fact that in METAFONT pixels can have more than two values.

5. METAFONT allows −.5s as such, nice syntactic sugar.

6. The difference is that in the first only the result is associated and stored under the name, while the latter stores and associates all that is between the curly braces with the name.

7. There are no scope braces.

8. For all the values supplied in the array the loop is executed.
The 2 \texttt{setlinejoin} prevents unwanted sharp corners. If you understand this code this note should be easy reading for you, and do contribute gems of your own.

1.2 Audience

The aimed at audience consists of users of (La)\TeX, METAFONT/Post, POSTSCRIPT, ... who are familiar with programming and not afraid of coding in terms of graphical primitives. The benefit of straight \texttt{POSTSCRIPT} (hand)coding is conciseness, efficiency, portability and universality. The drawback is little assistance, scarce diagnostic reports, tedious proofing, and so on, while developing the codes. Maybe the included codes can function as templates.

Reading (and understanding) all is too much hoped for. Nowadays we are flooded by information, and we all seem to suffer from the desease of our times: lack of time. If only one example will appeal to you or spark your imagination, I’ll be happy.

1.3 Why?

What has \texttt{POSTSCRIPT} got to do with \TeX?

EP history has it that \TeX and \texttt{POSTSCRIPT} form a real good team. \texttt{POSTSCRIPT} provides the graphical primitives which \TeX is lacking.

Some authors provide codes in BASIC, such as Lauwerier and Peitgen c.s. I completely agree that for those cases where images can be specified in terms of just a few graphical primitives it is wise to use omnipresent, cheap and trustworthy tools like \texttt{BASIC} (and Ghost/\texttt{POSTSCRIPT}), especially when time-invariance of the algorithms is at stake. However, with the ubiquitous \TeX together with multiplatform drivers which so easily and seemlessly merge the \texttt{POSTSCRIPT} pictures into the dvi script, one can go a step further and use \texttt{POSTSCRIPT} instead of \texttt{BASIC}, the more so because I expect \texttt{POSTSCRIPT} to enjoy a significant lifetime.

The above does not hold for other proposed teams like \TeX and SGML or PDF (yet) IMHO, with all respect.

1.4 Division of the plane

Related to tiling is the art of cutting a figure into pieces. And if we consider colors the stained-glass windows come to mind as beautiful examples.

---

Example (Stained-glass (monohedral) pattern)

In black-and-white the following are simple but famous examples.

Example (Lines and arcs)

Puzzles bridge the gap between cutting a picture into pieces – the design – and tiling – the remake.

2 Squares

A square is a common tile. An ordinary pavement is built from squares. Bathroom walls are tiled with them. And so on. Empty squares with deformed sides are abundant.

It’s amazing what can be achieved with just squares. Cut out the next one and tile, or better still switch on your computer and try some of your own.

Example (Squares with rounded corners)

The coding below illustrates how to tile in \texttt{POSTSCRIPT}: create a (path for a) tile, symmetrically around the origin.

---

9. In order to draw (free) forms it is better to name the pieces – triangle, square and diamond – and manipulate these. However, it is rather cumbersome to position them into patterns. For puzzling purposes it would be better if the X-works software provide the tangram pieces as part of their icons or allow a user to add icons. Considering the total process especially the reuse – in papers, as logos and so on – I’m happy with the investment of having coded the pieces in raw \texttt{POSTSCRIPT}.

10. For the codes see the Appendix.
and draw translated copies.

```plaintext
/r 10 def /a 50 def /ma a neg def /ha a 2 div def /mha ha neg def
/tile{4{ha r sub mha moveto mha mha mha mha r add r arcto
  90 rotate }repeat }
def
ma a a{/i exch def ma a a{/j exch def
gsave i j translate tile stroke grestore }
}for}for %showpage
```

Remarks. Programming the rounded corners via an appropriate use of `arcto` is borrowed from Adobe’s red book, the POSTSCRIPT reference manual. There are two parameters: the side of the square and the radius of the rounded corners. Note how the values of the hidden loop variable can be used.

In METAPOST the coding of the tiling template can read as follows.

```plaintext
a=50;r=5;
p; p:=quartercircle scaled 2r shifted (.5a-r,.5a-r);
p:=p--(p rotated 90);
tile:=p--(p rotated 180)--cycle;
for i=-a step a until a:
  for j=-a step a until a:
    draw tile shifted(i,j);
  endfor; endfor; showit; end
```

Remarks. Another possibility is to draw the tile and copy pictures. Note that only the corners are specified; the connecting straight lines are implicit.

**Example** *(Variant tiles)*

In art we have the tilings by members from ‘De Style’ like Mondrian and van Doesburg. But before we go over to Mondrian we will play a little longer with nearly empty tiles.

**Example** *(Puzzle)*

I leave it to your imagination how to draw the enclosed puzzle. It is fun to code the puzzle pieces. I did it with only two pieces, which of course are each others inverses in the sense that the sides fit. Nice exercise in using POST-SCRIPT’s symmetry operators. For the background any picture would do of course.

Patterns composed of these resemble the dragon figures as mentioned in The TExbook. In the following the tile is even further simplified by replacing the quarter circle by a straight line.

The above is obtained by the following code.

```plaintext
%!PS-Adobe- Truchet’s tiling, cgl Mrt 96
%!BoundingBox: -87.5 -87.5 87.5 87.5
/ma a neg def /ha a 2 div def /mha ha neg def
/tile{rand dup 2 idiv 2 mul eq {90 rotate}if
```

11. PostScript’s ‘for’ pops the value of the loop control variable on the stack among other things, such as executing the loop body. ‘/i exch def’ uses the value on the top of the stack and stores the value-name pair in the current dictionary, ready for later use of the loop control variable value by the name i.

The GUTenberg (French) TUG had in 1995 a contest along with their PSTricks/POSTSCRIPT tutorial about tiling ` a la Truchet. It turns out that for coloring 2 colors are sufficient. Have a try.

Example (Douat’s parquets)
A variant for parquets where a square is divided into 2 along the diagonal. By properly arranging the elements parquets are obtained. Douat\textsuperscript{15} classified varies parquets.

The left figure is obtained as follows.

\begin{verbatim}
%!PS-Adobe- Douat’s parquets, cgi April 97
%!BoundingBox: -37.5 -37.5 37.5 37.5
/a 25 def /ma a neg def
/ha a 2 div def /mha ha neg def
/qa ha 2 div def /mqa qa neg def
/ll{gsave qa mqa moveto
 mqa mqa lineto
 mqa qa lineto closepath fill
 qa mqa moveto
 qa qa lineto
 mqa qa lineto stroke
grestore}
def
/dotiling{f ma mul a f a mul{/i exch def
 f ma mul a f a mul{/j exch def
 gsave i j translate
tile stroke grestore
grestore}
def
/tile{gsave qa mqa translate ll grestore
 gsave mqa mqa translate ur grestore
 gsave qa mqa translate lr grestore
 gsave qa qa translate ul grestore}
def
for
for

Example (Lozenge by Mondrian)
\end{verbatim}


13. As communicated by Denis Roegel, see Cahiers GUTenberg 1995 for details. For solutions and the winner see the spring 1997 cahiers.
14. I solved the problem in POSTSCRIPT by looking at the colored pieces as pieces of a puzzle to be put together under the restriction of continuity of the colors along the sides. I grouped the 4 tiles into 2 sets. The tiling is obtained by taking a tile from each group on turns. Which tile to take from a group is free and therefore can be subject to throwing dice.
Remark. The real thing is not so perfect, it has lines of varying width making it more interesting. For computer art it might be a challenge to imitate this. But maybe we should leave the imperfections to humans and concentrate on what the computer is good at.

The POSTSCRIPT program below illustrates another approach to coding when the pattern is so straight: (redundant and clipped) lines, no tiles. Note the use of the symmetry by rotating over 90 degrees.

```postscript
%!PS-Adobe- Mondrian's Lozenge, cgl Jan 97
%%BoundingBox: -80 -80 80 80
/e 10 def /r e 8 mul def /mr r neg def
/frame{r 0 moveto 0 r lineto mr 0 lineto
    0 mr lineto closepath }
def
/lines{-7 1 7{e mul dup mr moveto
    r lineto }
    for stroke }
def
/diagonals{2 2 15{e mul dup r moveto
    neg mr exch lineto }
    for stroke }
def
gsave
/frame gsave .95 setgray fill grestore clip
lines diagonals 90 rotate lines diagonals grestore frame stroke %showpage
```

Example (Van Doesburg inspiration)

```postscript
%!PS-Adobe- Van Doesburg squares, cgl Feb 97
%%BoundingBox: 25 -6.25 250 115
/s 25 def /ms s neg def
/alfa s 3 div def /malfa alfa neg def
/h .5 s mul def /mhs hs neg def
/square{hs mhs moveto hs hs lineto
    mhs hs lineto mhs mhs lineto
    closepath stroke }
def
/frame{s mhs moveto 9 s mul mhs lineto
    9 s mul 4.5 s mul lineto
    s 4.5 s mul lineto closepath }
def
gsave frame clip
```

11(gsave 11(square s malfa translate)repeat
    grestore alfa s translate
)repeat grestore
/frame 3 setlinewidth stroke %showpage
```

Explanation. It is all about one square with appropriately shifted copies, with the shift parameterized by $\alpha$. Variant patterns can be easily obtained by changing the value of $\alpha$. The small squares are spurious. In reality van Doesburg enriched patterns like these by coloring the squares.

Example (Squaring the square)

Is it possible to partition a square into unique squares? A simpler problem is to divide a rectangle into squares, and allow double occurrences.16

```
16(gsave 11(square s malfa translate)repeat
    grestore alfa s translate
)repeat grestore
/frame 3 setlinewidth stroke %showpage
```

The following is a straightforward POSTSCRIPT code, with the use of (point) inversion. Interesting, and not trivial IMHO, is to write a POSTSCRIPT operator to draw the picture automatically starting with Bouwkamp's notational array as argument. A nice application of nested forall-s.

For the case at hand the notational array reads $[[6 4 5] [3 1] [6] [5 1] [4]]$.

```postscript
%!PS-Adobe- Squaring the square, cgl Mrt 97
%%BoundingBox: -75 -55 75 55
/square{ %s(ide) on stack
    /s exch def
    /hs .5 s mul def /mhs hs neg def
    mhs mhs moveto hs mhs lineto
    hs hs lineto mhs hs lineto
    closepath stroke }
def
30 square
2{gsave -45 25 translate 60 square grestore
    gsave 5 35 translate 40 square grestore
    gsave 50 30 translate 50 square grestore
    -1 -1 scale
}repeat %showpage
```

16. For an account of the squaring the square problem see for example Gardner’s More Mathematical puzzles booklet.
Example (Nested rectangles and spiral of life)

The nested rectangles have as ratio between width and height of the sides .618, that is an approximation of the golden ratio constant, commonly denoted by $\phi$. Nice is the (approximate) life spiral curve through the corners of the squares.

Example (Icons)

A central painting surrounded by smaller ones— in Russian: klema— illustrating episodes related to the central image.

In the code below the peripheral squares are the same, but can be varied of course.

Explanation. The tile—element—is parameterized over the size. The positioning is prescribed in the array indices. A double loop with an appropriate selection criterion whether a klema should appear. All that matters are the appropriate shifts and the coding of the element. The shifts are the values of the loop variables, apart from a scaling factor. The element is a framed star. The coding of the star has been treated in ‘Stars around I.’ It is rather straightforward, once you look upon it as a problem similar to the drawing of a polygon, namely with a broken line as side.

Example (2D regular surface)

This pattern is a remade of my youngest daughter’s handwork when at primary school. She did only one tile, that is a quarter of the pattern. This example illustrates the advantage of the computer. Once we have the basic element assemblages of translated or rotated copies can be done perfectly and easily.
Remark. Note how the spurious superellipse and cusp emerged as envelopes. A variant code more in the spirit of the earlier given template reads as follows.

```
% !PS-Adobe- Roos' 2D Gabo, cgl Dec 96
%%BoundingBox: -100 -100 100 100
/r 100 def /n 5 def /mn n neg def
/hr .5 r mul def /mhr hr neg def
/h hr n div def
/tile{2{mn 1 n{/y exch h mul def
hr y moveto y neg hr lineto
}for 180 rotate
}repeat
}def
mhr r hr{/i exch def
mhr r hr{/j exch def
gsave i j translate
i j mul 0 lt{90 rotate}if
tile stroke
grestore
}for
}for %showpage
```

METAFONT code. In this code use has been made of the operator point (expression) of (path).

```
size=50; path p;
p=origin..controls (0,.6) and (.4,1)..(1,1)
..controls (1,.4) and (.6,0)..cycle
&unitsquare) scaled size;
draw p;
for t:=1 upto 9:
% draw point .1t of p -- point 1+.1t of p;
draw point 2+.1t of p -- point 3+.1t of p;
draw point 4+.1t of p -- point 5+.1t of p;
endfor
addto currentpicture also
    currentpicture transformed
    (identity rotated 90 shifted (2size, 0));
addto currentpicture also
    currentpicture transformed
    (identity rotated 180 shifted (2size, 0));
showit; end
```

Remark. The inner boundary is part of the path because in the real thing the inside 'eye' was also filled with lines. These splines could have been specified equally simple in POSTSCRIPT, but there is as yet not a POSTSCRIPT equivalent of METAFONT's operator point of.

### 2.1 Groups of squares
Combining pentominoes – 5 squares grouped together – is a teaser.\(^{18}\)

**Example** *(Pentomino puzzle)*

The coding is tedious but straightforward.\(^{19}\)

---

18. Similarly one can group hexagons and so on as pieces for a puzzle.
19. As with the tangram, when we want to draw more tessellations it is convenient to write definitions for each piece and so on.
Because of the inherent symmetry the coding does not take much lines. Just a (rotated) square as basic element with rotated and translated copies to form the pattern. The diamonds, dodecagons and stars are spurious.

```plaintext
%!PS-Adobe- Square madness, cgl Dec 96
%!BoundingBox: -150 -150 150 150
/r 50 def
/s r 2.7320 div def
d r 2 mul s .707 mul sub def
/rotsq{gsave s 0 moveto
4{0 s lineto 90 rotate}repeat closepath stroke grestore}
dodeca{6{gsave r s sub 0 translate rotsq grestore 60 rotate}repeat}
gsave dodeca grestore
dodeca{6{gsave r s sub 0 translate rotsq grestore 60 rotate}repeat}

Explanation. The assembled squares yield spurious dodecagons. I chose as independent parameter for the drawing the radius \( r \) of the dodecagon. Its side equals \( \frac{r}{\sin \frac{75^\circ}{2}} \).

Note that the side of a square equals the side of the dodecagon. The radius \( s \) of the circumscribing circle of a square equals \( \frac{r}{\sqrt{3} \sin \frac{75^\circ}{2}} \). The shift \( d \) between dodecagons equals \( 2r - \frac{2s}{2} \), that is \( 2r \) minus half of the side, the overlap. Mandatory for the programming are: awareness of the CTM, the use of the graphics state scope delimiters \texttt{gsave} and \texttt{grestore}, and the difference between user and device space.

METAFONT code.

```plaintext
%!PS-Adobe- White squares, cgl Feb 97
%!BoundingBox: -65 -65 65 65
/r 50 def
/hs r 15 sin mul def
/s 2 hs mul def
/rotsq{hs 1 sub 0 moveto
4{90 rotate hs 1 sub 0 lineto}repeat closepath fill}
dodeca{12{gsave r 0 translate rotsq grestore 30 rotate}repeat}
gsave dodeca grestore
dodeca{12{gsave r 0 translate rotsq grestore 30 rotate}repeat}

Example (Dwirling squares)
Metamorphose or a circle limit? I associate it with the work of Schoonhoven. Well...what is in the name, who cares.

```plaintext
%!PS-Adobe- White squares, cgl Feb 97
%!BoundingBox: -65 -65 65 65
/r 50 def
/hs r 15 sin mul def
/s 2 hs mul def
/rotsq{hs 1 sub 0 moveto
4{90 rotate hs 1 sub 0 lineto}repeat closepath fill}
dodeca{12{gsave r 0 translate rotsq grestore 30 rotate}repeat}
gsave dodeca grestore
dodeca{12{gsave r 0 translate rotsq grestore 30 rotate}repeat}

Remark. Note that the picture can be assembled by shifts only. METAFONT does not allow to rotate pictures over arbitrary angles.

The following is left as a finger exercise for the reader.\(^{20}\)

20. By the way, it is interesting to see that the band of the right figure is enclosed from the outside by an 8-star and from the inside by a 6-star. However, the drawing can be done simpler.
Explanation. The square replaces a side of a dodecahedron, called ring here. Appropriately rotated and shrinked copies yields the result.

Example (Shrinking and rotated squares)
Lauwerier calls this a whirlpool.21

The left figure was obtained by the following code.

As can be seen from the above, I choose a hook together with an overlaced strip as basic element. This element is not symmetric. A tile is composed of 4 of these elements rotated over 90k°, for k = 0, 1, 2, 3, respectively. The resulting tile is (rotational, 4-valent) symmetric. Combining (translated) copies yields the pattern. The POSTSCRIPT code for the pattern above reads as follows, apart from the labeling.

How to draw these? What is to be considered as the basic element?
The coding in METAfont goes similarly, because pictures can be rotated over $90^\circ$, for $k = 1, 2, \ldots$

### 2.4 Square limits
Michel Goossens in his ‘LATEX en PostScript – de complementariteit in de praktijk’ has shown an example of this class of Eschers. Lengthy PostScript code borrowed from the net. The picture suffers from too much blackness near the boundary, IMHO, with all respect. This is an interesting aspect that we should draw thinner curves when they accumulate, because of the optical effect.

#### Example (Square limits grid)

![Square limits grid](image)

The code below is interesting because it demonstrates the use of recursion in POSTSCRIPT together with the use of the operand stack. Note that the line thickness also diminishes and that drawing is stopped overall at a certain depth, which I could not achieve via scale. Also interesting is the use of reflection about the line $y = x$. The code can be used as template for tiling—in the sense of Escher’s limit process—with scaled (and rotated) copies of a master tile created with the origin as center. Here the (contents of the) tile—element—is simply a cross.

```postscript
%!PS-Adobe Square limits grid, cgl Feb 97
BoundingBox: -96 -96 96 96
/gs 64 def /mgs gs neg def
/square{%lw s x y on stack
/y exch def /lx exch def
/hs exch 2 div def /mhs hs neg def
/lw exch def
gsave lx y translate lw setlinewidth
  element
grestore
setlinewidth .5 lw mul hs lx 1.5 hs mul add
  .5 lw mul hs lx 1.5 hs mul add
  y .5 hs mul sub
hs 1 gt {square square}
{8{pop}repeat} ifelse
def
/element{hs mhs moveto hs ha lineto
  mhs ha lineto mhs mhs lineto
  closepath stroke
  ha mhs moveto mhs ha lineto
  mhs mhs moveto ha ha lineto
  .25 llw mul setlinewidth stroke
}def
/pattern{4{/lw 1 def
  /s gs def /ms s neg def /x 0 def
gsave
6{gsave
  2{lws x x square
    -1 1 scale 90 rotate
  }repeat
grestore
  /s .5 s mul def /lw .5 lw mul def
  /x x 1.5 s mul add def
  repeat
grestore -90 rotate
  }repeat
}repeat
gsave -90 rotate
}repeat
/pattern %showpage
Escher enriched this grid with fishes for example. Maybe I’ll redo that example in due time.

#### Example (Variant with stars)

![Variant with stars](image)

22. The Sierpiński gasket—a fractal with limits allover—will be treated in the note on fractals.
23. On the net a larger code is available, of which the result is incorporated in Goossens’ article.
Example (Broideries)

Lauwerier introduced broideries. The tile is defined by a function with inherent symmetries. Lauwerier used among other things reflection symmetric around the x-axis, y-axis, and the lines $y = \pm x$, for example as in the function $f(x, y) = (1 - x^2)(1 - y^2)$. The idea of broideries are obtained if at a point $(x, y)$ a mark is drawn when the function value satisfies a certain criterion. In the broidery below a mark–small cross–is drawn at $(x, y)$ if $\text{int}(400 f(x, y))$, equals a multiple of 3, for $x = -1, \ldots, -1/n, 0, 1/n, \ldots, 1$, $y = -1, \ldots, -1/n, 0, 1/n, \ldots, 1$. $n = 40$ was used.

3 Triangles

Triangles tile well, especially in space. For example they are used as elements in approximating surfaces among other things in analogy with line pieces for approximating curves in the plane. They form the sides of the Platonian solids tetraeder, octaeder, and icasohedron. I’ll restrict myself to triangles in the (xy-)plane.

Example (Puzzle of groups of triangles)

As with tangrams we can write definitions for the pieces and so on.

Example (Flexed triangles and coronas of triangles)

We can modify the sides or arrange them in patterns.

The codes for these drawings read as follows.

\%!PS-Adobe- Flexed triangles, cgl Feb 97
\%%BoundingBox: -15 -15 115 115
/s 50 def
.hs .5 s mul def /mhs hs neg def
/hss 1.732 hs mul def
/qs .5 hs mul def /mqs qs neg def
/r 1.15 hs mul def
/hr .5 r mul def /mhr hr neg def
/tri(mhs mhr moveto
3(mqs mhr qs add
qs mhr mqs add hs mhr curveto
120 rotate )repeat fill
)def
tri
gaave 2{s 0 translate tri}repeat grestore
gaave hs hss translate tri
s 0 translate tri
grestore
s 2 hss mul translate tri %showpage

I did not rotate to allow more easily extension of the pattern.

\%!PS-Adobe- Coronas of triangles, cgl Feb 97
\%%BoundingBox: -72.5 -72.5 72.5 72.5
/r 50 def /r(1+1.732 sin 15)
.hs r 15 sin mul def /mhs hs neg def
/hss 1.732 hs mul def
/f r r hs add div def
/tri(hss 0 moveto
0 hs lineto 0 mhs lineto
closepath fill
)def
3{12{gsave r 0 translate tri grestore
30 rotate
}repeat f f scale 15 rotate
}repeat %showpage

Note the use of scale.
Example (Non-tight) triangles

For the coding the same template as for the square tiles was used. Note the spurious squares, stars and octagons.

Example (Triangular arms: east-west rencontre)

With the following (tail) recursion code.

```
%!PS-Adobe- Triangle arms, cgl Feb 97
%%BoundingBox: 0 -50 185 50
/s 64 def /ts 2 s mul def
/tri{/locs exch def
  0 locs moveto locs 0 lineto 0 0 lineto
closepath currentpoint stroke
translate -45 rotate
/locs .707 locs mul def
locs 1 ge {locs tri} if 
}def
2 setlinejoin
%!
/gsAVE s tri grestore
ts ts translate 180 rotate
s tri %showpage
```

Example (Graph of icasohedron)

The icasohedron has triangles as sides. The following projection shows them all. ‘Tiling’ to assist insight.

The left projection is obtained as follows.

```
%!PS-Adobe- Graph icasoheder, cgl Mrt 97
%%BoundingBox: -50 -30 50 60
/s 100 def /ss 1.732 s mul def
```

Example (Nested triangles)

This triangle comes back in pentagons: a side with two diagonals. The golden ratio is in there, and we can split of similar smaller triangles infinitely. The code is in the same spirit as that for rectangles.

Example (Triangular limits grid)

In analogy with the square limits grid there is the triangular limits grid, which is related to the Sierpiński carpets, let us say it is an outbound variant. I did not find this one in Escher’s work. Maybe too much complexity in the corners?

The code is in the same spirit as the code for the limit squares. ²⁴

²⁴ The Sierpiński gasket – a fractal which is limiting allover – will be treated in the note on fractals.
4 Pentagons

Pentagons don’t tile tight. So what?

Example ((Non-tight) pentagons I; lips)

In the coding the reflection via scale is interesting.

```
%!PS-Adobe- Simple Pentas, cgl Feb 97
BoundingBox: -175 -95 175 95
/r 50 def /mr r neg def
/rin r 36 cos mul def
/mhdx s 18 cos mul neg def
/pentagon{r 0 moveto
5{72 rotate r 0 lineto}repeat
stroke}def
%figure{pentagon}def
2{gsave mhdx mr add 0 translate
figure
2{gsave 36 rotate 2 rin mul 0 translate
figure
stroke 1 -1 scale
repeat
stroke -1 1 scale
repeat %showpage

dürer tiled straightforwardly as follows. Only spurious diamonds show up.
```

Example ((Non-tight) pentagons II)

Because of the inherent symmetry the coding does not take much lines. Just a pentagon as basic element with rotated and translated copies. The diamonds are spurious. Do you see how to code another ring, how to let the pattern grow?

```
%!PS-Adobe- Duerer, cgl Jan 97
BoundingBox: -55 -65 55 65
/r 15 def /rin r 54 sin mul def
/p{gsave r 0 moveto
5{72 rotate r 0 lineto}repeat
stroke grestore
}def
/tr{2 rin mul 0 translate}def
p 36 rotate
5{gsave tr
p 36 rotate tr p grestore
gsave -36 rotate tr p grestore
grestore 72 rotate}repeat %showpage

METAFONT code. My earlier METAFONT code did not take into account the fact that the diamonds are spurious. Part of it is given below. Remarkable is that use is made of reflectedabout which is not needed in the simpler approach as worked out in the later POSTSCRIPT code.

```
r=15; n=5; s=2r*sen36; path p[];
diag=s*[1+2sen18];
height=s*[cos54+cos18];
pl=(r,0)--(r*cos18, s*sen18);
p2=(r,0)--(r+s*sen18, s*sen18)
--(r+s*sen18, s*(1+sen18))
--(r, diag)--(r*cos18, r*sen18);
p3=(r+s*sen18, s*sen18)
--(r+2s*sen18, s*sen18)
--(r+s*sen18, s*(1+sen18));
p41=(r+2s*sen18, 0)
--(r+s*sen18, -s*sen18);
p42=p41 reflectedabout
(origin, (r+s*sen18, s*(1+sen18)));
for k=72step2 until 360:
draw p1 rotate k;endfor
for k=0step2 until 360:
draw p2 rotate k;endfor
for k=0step2 until 360:
draw p3 rotate k;endfor
for k=0step2 until 360:
draw p4 rotate k;endfor
showit; end
Example *(Decahedron at the heart)*

5 Hexagons

Hexagons tile tight and may yield intriguing patterns. Notice the difference in the optical effect.

Example *(Hexagon template tilings)*

POSTSCRIPT code. A hexagon is underneath, that is a cube in projection. Starting from this the code is straightforward.

Example *(Horak’s cubes)*

An optical effect is obtained, it looks like projected cubes. The minimal information—the parameter of the figure—is the radius $r$ of the circumscribing circle of the hexagon.

Najaar 1997
METAFONT code.

s=40; path p[]; %cgl Jan 97
p1=origin--(0,2s);
p2=p1 rotated-120 shifted(0,s);
p3=p1 & p2 shifted(0,s);
p4=p1 shifted(1.732s,-s);
for k=0 step-120 until-240: %draw tile
draw p2 rotated k;
draw p3 rotated k;
draw p4 rotated k; endfor
for k=1upto2: %draw pattern
  addto currentpicture also currentpicture
    shifted (k*(1.732s,3s));
  addto currentpicture also currentpicture
    shifted (3.464s*k,0);
endfor
%Cut out a piece
cullit;
fill unitsquare xscaled 10 size
  yscaled (7.5 size shifted (2 size,0));
cull currentpicture keeping (2,infinity);
%Frame the piece
pickup pencircle scaled 3;
draw unitsquare xscaled 10 size
  yscaled (7.5 size shifted (2 size,0));
showit; end

Remarks. The clipping must be done differently in META-
FONT, because the concept of a clipping boundary is not
there. POSTSCRIPT allows any path. METAFONT requires
as path a continuous line. As a consequence I have split up
the basic element, although I could have traversed various
parts of the element more than once.

Example (Variant)

Example (Romanovsky’s Chinese porcelain)
A broderie of overlapping hexagons. The minimal infor-
mation required to draw this pattern is the height s and
the width w of the basic bar.

POSTSCRIPT code.

%!PS-Adobe-
% Romanovsky’s porcelain, cgl Jan 97
%!BoundingBox: 0 0 255 180
/s 30 def /ms s neg def /ss s 1.732 mul def
/h s 2 div def /mhs hs neg def
/hss hs 1.732 mul def
/w s 3 div def /wds w 1.732 div def
/hw w 2 div def
/hws hw 1.732 div def
/line(hw s hws sub moveto hw hws lineto)
def
/tile3{gsave line -1 1 scale line stroke
grestore 120 rotate
grestore 3 repeat}
def
/pattern{gsave
  6{gsave
    7{gsave tile
    hw has add hs hws add translate tile
    grestore w ss add 0 translate
  }repeat
  grestore 0 s wds add translate
}repeat grestore}
def
/6{gsave
  7{gsave tile
    hw has add hs hws add translate tile
    grestore w ss add 0 translate
  }repeat
  grestore 0 s wds add translate
}repeat grestore}
def
/gsave frame clip newpath
  mhs mhs translate pattern
grestore
/frame 3 setlinewidth stroke %showpage

Explanation. As height I took the radius of the circum-
scribing hexagon. The tile is a hexagon, with 6 inner line
pieces. The basic element is drawn by exploiting fully its
symmetry. Once the shifts have been calculated the coding
is straightforward. The pattern results after first adding a
copy translated by (s√3 + w, s + w/√3)/2, and then tile
pointtopair projection operators.
with horizontal and vertical translations $k(s\sqrt{3} + w, 0)$, and $k(0, s + u/\sqrt{3})$, for $k = 1, 2, \ldots$.

METAFONT code. Note that in METAFONT we can’t rotate pictures arbitrarily, but we can do with paths. The code below I wrote a year earlier than the POSTSCRIPT code, and differs a little, if not for the basic element.

```plaintext
s=30; w=.333s; path p[]; %cgl, Jan 97
p1=( .5w, .288w)--( .5w,s-.288w);
p2=(-.5w, .288w)--(-.5w,s-.288w);
for k= 0 step 120 until 240: %tile
  draw p1 rotated k; draw p2 rotated k;
endfor
addto currentpicture also currentpicture shifted(.866s+.5w,.5s+.2887w);
for k= 1 step 1 until 2:
  addto currentpicture also currentpicture shifted(k*(0, s+.5774w));
  addto currentpicture also currentpicture shifted(k*(1.732s+w,0));
endfor
cullit;%clipping boundary functionality
fill unitsquare xscaled 6.5s yscaled 4s shifted(.25s,0);
cull currentpicture keeping (2,infinity);
pickup pencircle scaled 3; %frame
draw unitsquare xscaled 6.5s yscaled 4s shifted(.25s,0);
showit; end
```

Example (Variant hex pattern)
And what about the following with only the side of the hexagon as parameter?

![Hexagon pattern](image)

6 Escher

M C Escher, a Dutch (graphics) artist of the first half of the XX-th century, has among other things enriched classical tilings by loosening the strict geometrical tradition. He drew creatures like Buddhas, reptiles and so on, appealing to the masses, well … science biased.

Similar to the above classical plait his basic element is not symmetric.26

He realized that the picture on the tile is only restricted by the points where it cuts the boundary. Maybe he just thought of deforming the boundaries. Moreover, the boundaries are related by symmetry, induced by tiling.

Example (Escher-like fishes)
The following is borrowed from Lauwerier’s earlier mentioned booklet ‘Symmetrie etc.’ The tiling is much in the spirit of our earlier template. The basic tile consists of 2 lines, one also translated and the other also reflected. The middle row consists of reflected tiles: fishes swimming in the other direction.

![Fishes tiling](image)

6.1 Escher’s Squares

Example (Escher’s Buddhas)

How to draw these? How to program?

Analyzing the picture reveals that the basic element is a modified side of the square. This side can be rotated and reflected to yield a square. Rotated copies of this square form a tile. Translated copies will yield the pattern.

Below the basic element has been simplified into a line with a notch, and the intermediate phases have been shown.

26. MacGillavry, a Dutch crystallographer, has studied Escher’s symmetries and published on the issue.
Example (Escher’s mechanism)

Programming the four phases

\[ element \xrightarrow{R,M} square \xrightarrow{R} tile \xrightarrow{T} pattern \]

can be read from the following POSTSCRIPT code for Escher’s Buddhas. (\( R, M, T \) denote Rotation, Mirroring and Translation, respectively.)

```postscript
%!PS-Adobe- Escher Buddhas, cgl Dec 96
%%BoundingBox: -20 -45 320 140
/s 20 def /ms s neg def /ts s s add def
/pa{s s}def
/pb{.7 s mul 1.7 s mul}def
/pc{-.4 s mul 1.45 s mul}def
/pf{.35 s mul s}def
/ankle{-.1 s mul .6 s mul}def
/toe{0 .3 s mul}def
/heel{-.4 s mul .6 s mul}def
/head{-.9 s mul .9 s mul}def
/center{-.25 s mul 1.8 s mul}def
/cpa{0 1.55 s mul}def
/cpb{.25 s mul 1.55 s mul}def
/cpc{-.6 s mul 1.1 s mul}def
% /element{gsave
pa moveto pb pb finger curveto
pc moveto cpa cpb pd curveto
knee knee pf curveto %pf lineto
ankle lineto
toe lineto
heel lineto
center .75 s mul 315 225 arcn
heel moveto cpc cpc head curveto
stroke grestore}def
%
/tile{element
gsave 90 rotate 0 s -2 mul translate
element
grestore gsave 1 -1 scale 90 rotate
element
grestore gsave -1 1 scale
0 s -2 mul translate
element
grestore gsave ms s moveto s ms lineto
.05 setlinewidth stroke grestore
}def
%
/pattern{gsave
tile s 4 mul 0 translate
tile 0 s 4 mul translate
tile s -4 mul 0 translate
tile grestore
}def
%
/as 8 s mul def /mas as neg def
/contour{ms ms moveto
as 0 rlineto 0 as rlineto
mas 0 rlineto closepath
}def
%
/Times-Roman findfont
10 scalefont setfont
%Draw
element
ms ms 25 sub moveto (element)show
65 0 translate square
ms ms 20 sub moveto (square)show
75 0 translate tile
0 ms 20 sub moveto (tile)show
110 0 translate
0 ms 20 sub moveto (clipped pattern)show
contour clip pattern
contour 2 setlinewidth stroke %showpage

My METAFONT code is similar to the code for the reptiles given below.

6.2 Escher’s hexagons
The same mechanism as demonstrated with squares was applied by Escher to a hexagonal grid.

Example (Reptiles)
METAFONT code.\textsuperscript{27}

\% Escher's Reptiles, cgl May 96
pickup pencircle scaled 1;
pair p[ ]; path ctoad;
s:=50;
% hexagon p1--p3--p6--p8--p10--p12--cycle
% as grid/canvas
p3=s*up;
p6=p3 rotated60;
p8=p6 rotated60;
p10=p8 rotated60;
p12=p10 rotated60;
p1:=p12 rotated60;
% pickup pencircle scaled .05;
% draw p1--p3--p6--p8--p10--p12--cycle;
p2=3333{p3,p1};p13=3333{p12,p1};
p4=3333{p3,p6};p7=3333{p8,p6};
p5=3333{p6,p3};
p9=3333{p10,p8};
p11=3333{p10,p12};
% pickup pencircle scaled 5;
% for k=1 upto 13: drawdot p[k]; endfor
p14=.166667{p3,p6} + .1s*down;
p141=p14 + 1.1s*(-.17,.1);
p15=.25{p3,p10};
p16=.166667s*(-1,1);
p17=p16 + .3333s*up;
p20=p19 + .3333s*down;
p21=p18 + s*(.28,0);
p22=.22{p9,p2};
p23=.31{p11,p4};
p24=p12 + 1.5s*(-.17,.1);
p25=.24-.1s*(1.5,5);
p26=25+.1s*left;
p27=p26+.3333s*up;
p28=.2{p13,p6};
pickup pencircle scaled 2;
% a complete toad, well reptile
ctoad=p1--(p1--p14--p141--p2)
rotatedaround(p1,120)--
(p8--p21--p9)rotatedaround(p1,-120)
shifted (0,-3s)--
p11--p23--p22--p10--
(0--p22--p23--p11)
rotatedaround(p10,120)--
p9--p21--p8--
(p3--p15--p16--p17--p4)
rotatedaround(p1,-120)
shifted (-1.5s*sqrt3,-1.5s)--
p7--p20--p19--p18--p5--p6--
(p6--p5--p18--p19--p20--p7)
rotatedaround(p6,120)--
p4--p17--p16--p15--p3--
rotatedaround(p1,-120)--
p2--p141--p14--p1;
draw ctoad;
draw ctoad rotatedabout (p1,120);
draw ctoad rotatedabout (p1,-120);
showit; end

Explanation. This code is based on points on the circumference of the hexagon. These are related by symmetry induced by the tiling as can be seen from the figure. A more extensive pattern can be obtained by adding shifts to the current picture.

POSTSCRIPT code by which the included figures are drawn.

\%
/Escher's reptiles, cgl Jan 97
%boundingBox: -50 -70 130 150
/r 50 def
/mr r neg def
/hr r 2 div def
/mhr hr neg def
/hrx hr 1.732 mul def
/mqr qr neg def
/er r 8 div def
/mer er neg def
/gxx x 2 div def
/merx erx neg def
/delta er 2 div def
/head{hr 0 moveto
qr mqr 1.25 mul lineto
0 mqr 1.25 mul lineto
mqr 0 lineto
mqr qr 1.5 mul lineto
merx er lineto
mhr 0 lineto
stroke
}def
/sides{hr 0 moveto
mer 2 div delta sub
mhr delta sub lineto
delta mer lineto
merx er lineto
0 hr er sub lineto
mhr erx add er lineto
mhr 0 lineto
stroke
}def
/tail(hr 0 moveto
qr mhr lineto
mqrx mer lineto %end

Explanation. This code is based on the symmetry of pairs of sides of the hexagon: head, side and tail. Open as yet is how to compose contours from these for coloring purposes.

6.3 Circles
With circles we have two special locations: the centre and the boundary. I associate this with implosion and explosion. When convergence is towards the centre it is about a centre of multiplication. When convergence is outbound I’ll talk about a circle limit, in pursuit of Escher.

Example (Disk divisions)
Example (Windows: From a barn and from Malbork)

6.4 Circle limits

Escher also used hyperbolic grids: circular arcs within a circle, where the centers of the circles converge towards the boundary, and the arcs cut the boundary perpendicularly.

It did take me some time to realize what is really meant by circle limits. The basics, unblurred by Escher’s approach, is really simple, and in analogy with triangular and square limits, and ... as it turned out easier to code as well. Let us first do it à la Escher followed by a straight one.

Example (Circle limits grid à la Escher)

Escher extended this also to the surface of a sphere, for example in his ‘Angels and Devils.’

Example (The straight circle limits grid)

In analogy with squares and triangles the straight circle limits are even easier to code. The starting point is the starting number of arcs n as parameter. 360 divided by n yields the arc of the main circle cutout by the biggest circle of the limit set of circles. The centre and radius of the circle follow easily from the data of the main circle, the cutout arc, and that the circles intersect orthogonally. Rotate the template – ring – and loop this levels times. Note that en-passant the line thickness is decreased to counteract the blackening effect.

Najaar 1997
Circles within a circle  Apollonius already considered circles which touch on 3 circles.  

The equations for a circle, in terms of its centre and radius \( \{x, y, r\} \), which touches on 3 other circles are

\[
(x - x_i)^2 + (y - y_i)^2 = (r_i \pm \delta)^2, i = 1, 2, 3
\]

with \( \{x_i, y_i, r_i\} \) the data of the given circles. \( \pm \) accounts for touching on the outside, respectively inside.  

For the covering below the 3 quadratic equations can be simplified into 2 linear equations and a quadratic equation. These linear equations have been made explicit as function of \( \delta \) for the centres of circles which touch the main circle and the circumscribing circle.

Example (Circle covered by circles)

Explanation. I chose as origin the midpoint of the circumscribing circle. The main circle has radius \( R \) and the circumscribing circle has radius \( R + r \).

Because of this special configuration, next to my choice of origin and x-axis, the equations for the centre of the circle which touches the outer circle and the main circle, parameterized over its radius \( \delta \), simplify into

\[
\begin{align*}
x &= R + r - \frac{2R + r}{r} \delta \\
y^2 &= (R + r - \delta)^2 - x^2.
\end{align*}
\]

Limit situations are \((x, y) = (R + r, 0), (-R, 0)\), for \( \delta = 0, r \).

The requirement for touching the 3rd circle, \( \{x_i, y_i, r_i\} \), yields the equation in \( \delta \), with \((x, y)\) given by the above formulas

\[
(x - x_i)^2 + (y - y_i)^2 = (r_i + \delta)^2.
\]

By changing the meaning of the 3rd circle repeatedly – the just determined circle becomes the next – the set of shrinking touching circles, suggestively infinitely, has been obtained.

The above considerations and ideas have been coded in POSTSCRIPT as follows.

%!PS-Adobe- Circle with circles, cgl Mrt 97
%%BoundingBox: -150 -150 150 150
/R 100 def /r 50 def /Rr R r add def
/xi R neg def /yi 0 def /ri r def
Rr 0 moveto r 0 R 0 360 arc
Rr 0 moveto 0 0 Rr 0 360 arc
r neg 0 moveto xi yi r 0 360 arc
stroke
%tangent circle: x, y, d(elta)
x{Rr d 2 R mul r div 1 add mul sub}def
/y{Rr d sub dup mul x dup mul sub sqrt}def
/xd{ri d add
x xi sub dup mul
y yi sub dup mul add sqrt sub}def
/solveit(/invariant: 1 fd > 0 and u fd <= 0
/d .5 1 u add mul def
fd 0 ltl/1 d def)
{/u d def}ifelse
u 1 sub eps gt{/solveit
}ifelse
)
def
%

28. The concise and implicit equation for \( \delta \) is suited for solving by computer. The explicit solution was communicated by H.J. van de Stadt and is due to Soddy. Another approach for obtaining an explicit formula for \( \delta \) in terms of a square root and the arithmetic operations is by formula manipulation programs. I’m not sure whether these programs would yield Soddy’s elegant representation.
Remarks.
I coded a—primitive, quick and dirty—zerofinding (bisection) operator in POSTSCRIPT. 29
Interesting is to extend tiling to the hyperbolic plane à la Coxeter, or to variate the circular arc by a pleasing line, for example anthropomorphically à la Escher.
To write a program to fill up all (3-sided) cusps by circles, or all the new circles, infinitely, is interesting, and a nice, but not trivial, exercise in recursive programming. 30
An explicit solution for the radius of the 4th inscribed circle due to Soddy was communicated when the paper was in proof by H.J. van de Stadt and reads as follows.

\[ \frac{1}{\delta} = \frac{1}{r_1} + \frac{1}{r_2} + \frac{1}{r_3} + 2 \sqrt{\frac{1}{r_1 r_2} + \frac{1}{r_2 r_3} + \frac{1}{r_3 r_1}} \]

I decided to let the POSTSCRIPT code for this drawing unaltered. The use of the zero finding in POSTSCRIPT for solving an equation might be of use in other contexts.

Pondering about this problem made me realize that times have changed. We don’t need anymore for these kinds of problems solutions by compass and ruler, or solutions by transformation techniques such as the inversion in a circle. 31 With our fancy computers approaches akin to this tool are emerging, and IMHO, with all respect, the above is such an approach. Apollonius revisited, aha.

\[ \text{Example (The old approach)} \]
For the interested reader a picture which exhibits the inversion method is included on the left. The bold circles are what we are talking about. The thin circles and lines are the inverted ones, with the dashed circle the circle of inversion.

\[ \text{Example (Squares limits variant with circles)} \]

6.5 Metamorphoses
Escher played with changing the patterns while tiling. Schröfer did this with changing a circle and his result is captivating, with a nice Op Art effect. V asarely 32 has exploited this in depth, and used colors with breathtaking results.

Hofstadter 33 calls this parquet deformations.

\[ \text{Example (Schröfer’s Op Art)} \]
Note how the circle in the center grows when moving out bound, and in general becomes an ellips. The vertical and

29. Maybe, I should recast in POSTSCRIPT in due time the famous zeroin algorithm developed at the CWI, where a combination of strategies—bisection, interpolation and extrapolation—have been implemented.
30. But, . . . maybe there is another inroad when we look upon it as a fractal. H.J. van de Stadt mentions that this fractal is known as Pharaoh’s Breastplate, and can be found in the book on fractals by Mandelbrot. In the note on fractals I’ll come back on Soddy’s formula and the fractal.
32. A Hungarian computer artist.
33. Hofstadter, D R (: Metamagical themas—questioning for the essence of mind and pattern.
horizontal axes of the ellipses are proportional to the absciss
and ordinate of its position.

Example (Division of hexagon and octagon)

Just diamonds, a star or cubes? Isn’t it nice that hexagons
can be divided so beautifully into 12 diamonds?

Remark. In the coding the use of an array for the indices,
next to \texttt{forall}, and the flipflop mechanism are interesting.
My \texttt{METAFONT} code is similar, modulo some syntactic
sugar.

Example (Variants)

7 Tiling by diamonds

The following example allows a nice demonstration of how
to use the ‘Turtle graphics’ idea for coding in \texttt{POSTSCRIPT}.
Remark. It’s interesting to write a program which generates automatically tilings by these kites and darts. That is to teach a computer how to puzzle.

**Example** *(Rhombus as basic element)*

I chose a diamond as basic element. Of course one can also code the wiggy lines.

![Diagram](image1)

The above is obtained as follows

```latex
%!PS Tiling fourthree, cgl 96
%%BoundingBox: -100 -100 100 100
/a 10 def /ha a .5 mul def
tile {% rhombus + 90 rotated rhombus
ha 3 sqrt mul 0 moveto
0 ha lineto ha 3 sqrt mul neg 0 lineto
0 ha neg lineto closepath
ha 1 3 sqrt add mul ha 3 sqrt mul lineto
ha 2 3 sqrt add mul 0 lineto
ha 1 3 sqrt add mul ha 3 sqrt mul neg lineto
closepath
}def
tena a 10 mul def
/frame {tena neg tena moveto
tena 2 mul 0 rlineto 0 tena -2 mul rlineto
tena -2 mul 0 rlineto closepath}def
dotiling{a -11 mul tena neg translate
9{gsave
11{tile a 1 3 sqrt add mul 0 translate
}repeat stroke
grestore
gsave ha 1 3 sqrt add mul dup translate
11{tile a 1 3 sqrt add mul 0 translate
}repeat stroke grestore
0 a 1 3 sqrt add mul translate
}repeat
}def
```

My METAFONT code reads similar.

### 8 Tiling by stars

The following – non-tight ‘hexagon’ tilings – reminds me of tilings by the Moors.

![Diagram](image2)

```latex
%!PS-Adobe 6-stars, cgl Feb 97
%%BoundingBox: -70 -70 70 70
/r 25 def 2 setlinejoin
/six{%r on stack
/rloc exch def
six{rloc 0 moveto
6{30 rotate .577 rloc mul 0 lineto
30 rotate rloc 0 lineto}repeat closepath stroke
}def
30 rotate r six .588 r mul six
6{gsave 2 r mul 0 translate
r six .5 setlinewidth .588 r mul six
grestore
60 rotate}repeat %showpage
```

Looking at the left picture I found it hard to recognize that it was just a tiling of 6-stars. In reality the tessellations also exhibit ‘hidden’ lines.

**Example** *(Dual of composition of pentagons)*

Stars and the enveloping polygons are related.

As exercise for the reader the following dual of the composition of pentagons.

![Diagram](image3)
This code arose after Jackowski’s visit to NTG’s fall 1996 meeting, where he lectured and dealt with stars among other things.

Explanation. The figure is composed of rings of stars, with in each ring stars of 2 sizes. The bigger stars circumscribe a spurious pentagon. The radius $r$ of (the circumscribing circle of) this pentagon is taken as the independent parameter of the drawing. Dependent quantities are the following.

Radius inner circle: $r_{\text{inside}} = r \cos 36$

Side pentagon: $2r \sin 36$

Bigger star
radius: $r \cos 36$

distance from center: $2r \cos^2 36$

Smaller star
radius: $0.5r$

distance from center: $r_{\text{inside}} + 0.5r$

The radius of the next ring: $r \cos (2r \cos 36 + 1)$

Bounding Box: $r \cos (2r \cos 36 + 1))^{1/2} \approx 2.12r$.

Drawing a star has been explained in ‘Stars around I.’

Note the use of `2 setlinejoin` to circumvent spurious sharp corners.

Example (Variant)

9 Acknowledgements

Thank you Bogusław Jackowski for your suggestions, inspiring examples, and help. Thank you Sasha Berdnikov for providing me with examples of tilings. Thank you Erik Frambach for comments and discussions. Peter van Summeren pointed me to some literature items. Denis Roegel informed me about Truchet’s tiling and the GUTenberg

35. Radius of circumscribing pentagon, with $n$ the number of rings.

36. See MAPS 97.1. The idea is that the star can be seen as a pentagon with a broken line as side. The coordinates of the junction of the 2 line elements of the broken line follow from a backside of the envelope calculation.

37. Courtesy Marcel Tünissen, who also played with the figure and applied technology from 3D–extending the sides of polyhedra to yield starred figures – to 2D.
contest on the issue, next to providing me with a copy of a GUTenberg article about the winners of the contest. H.J. van der Stadt communicated Soddy’s formula.

Many examples have been borrowed from the earlier mentioned book by Grünbaum and Shephard.

As usual Jos Winnink proofed the paper and lent a helping hand in procrusting the article into MAPS outfit. Thank you all.

10 What more?
The tilings, as overwhelming they might seem, are just the top of the iceberg. What about tiling on curved surfaces, for example on a sphere, as well as general tiling in 3D? Maybe, I will come back on the issue when computers have real holographic 3D viewing possibilities.

11 Conclusions
Just the use of a few graphical primitives in POSTSCRIPT and awareness of the user versus device space can yield interesting pictures via little POSTSCRIPT code. Sometimes it is handler and more elegant to use METAFORENT as a declarative language. On the other hand METAFORENT’s pictures can’t be rotated over arbitrary angles.

Differences in the concept of the path data structure in POSTSCRIPT and METAFORENT entails different programs. The most apparent differences are in tiling and clipping. I would welcome in POSTSCRIPT an operator similar to METAFORENT’s point expression of path.

Happily the same splines are used underneath facilitating METAFORENT to calculate the splines from a declarative specification ready for use in POSTSCRIPT. I never realized that tiling, especially when done by computer, can be so amusing. So far I’ve no access to color printers, alas. Undoubtedly a mer a boire awaits me there.

I hope that the given codes will contribute to the literature of METAFORENT and POSTSCRIPT codes. I welcome comments.

12 Appendix: Postponed codes
12.1 Mondrian’s Two lines
In the original Mondrian the two lines did not divide the sides by the golden ratio. An oversight by Mondrian?

In principle a very simple code, but becomes interesting with proper coloring of the background and with lines of non-negligible thickness.

%!PS-Adobe- Mondrian’s Two lines, cgl 96
%%BoundingBox: -50 0 50 100
/D 50 def
/R D 1.4142 mul def %side
/gr .618 R mul def
/delta 10 def /mdelta delta neg def
45 rotate
0 0 moveto R 0 lineto
R R lineto 0 R lineto
closepath
gsave .985 setgray fill grestore
clip newpath
%gr 0 moveto 0 gr lineto
gr delta add mdelta moveto
mdelta gr delta add lineto
%gr R moveto 0 R gr sub lineto
gr delta add R delta add moveto
mdelta R gr sub delta sub lineto
R 30 div setlinewidth stroke %showpage

Explanation. I chose for a clipping boundary and to let the thick lines extend to be cut appropriately.

12.2 Yin-Yang
An exercise in using (circular) arcs.

METAFORENT code.

%Yin-Yang, cgl Jan 97
size=100;
fill ((halfcircles
 halfcircle rotated 180 scaled.5 shifted(-.25,0)&
 reverse (halfcircle scaled.5 shifted(.25,0))&
cycle)scaled size);
draw fullcircle scaled size;
fill fullcircle scaled .125size
 shifted (.25size, -.0625size);

Najaar 1997
unfill fullcircle scaled .125size
    shifted (-.25size, .0625size);
showit; end

Explanation. It is all about drawing, casu quo (un)filling, of (arcs of) circles, and to scale and reposition these appropriately. In plain METAFONT we have half/fullcircle which have to be scaled, rotated, and positioned. In POSTSCRIPT a circle macro is easily written and used similarly. Interesting is the use of reverse in METAFONT in order to create closed contours, which in POSTSCRIPT is done implicitly via arc, that is the arc in negative direction.

A similar POSTSCRIPT code reads as follows.

%!PS-Adobe- Yin-Yang, cgl 96
%%BoundingBox: -50 -50 50 50
/R 50 def /mR R neg def
/r R 8 div def /mr r neg def
/%circle%@center on stack
translate
r 0 moveto 0 0 r 0 360 arc
}def
R 0 moveto 0 0 R 0 180 arc
mR 0 hR 180 360 arc
mr 0 hR 180 0 arcn
fill
gsave mhR r circle
1 setgray fill grestore
gsave hR mr circle
0 setgray fill grestore
R 0 moveto 0 0 R 0 360 arc stroke %showpage

The following POSTSCRIPT code was output by MetaPost.38 Some ≈ 50 lines of curveto, fill and stroke. Much less readable than my straight (handcoded) PostScript, moreover, ‘structure’ has disappeared.

%!PS-Adobe- Yin-Yang, MetaPost (JJW)
%%BoundingBox: -51 -51 51 51
newpath 50 0 moveto
50 13.26 44.73 25.98 35.35 35.35 curveto
25.98 44.73 13.26 50 0 50 curveto
-13.26 50 -25.98 44.73 -35.35 35.35 curveto
-44.73 25.98 -50 13.26 -50 0 curveto
-50 -13.26 -44.73 -25.98 -35.35 -35.35 curveto
-25.98 -44.73 -13.26 -50 -50 0 curveto
13.26 -50 25.98 -44.73 35.35 -35.35 curveto
44.73 -25.98 50 -13.26 50 0 curveto
closepath stroke
newpath 31.25 -6.25 moveto
-18.75 6.25 moveto
-18.75 7.91 -19.41 9.50 -20.58 10.67 curveto
-21.75 11.84 -23.34 12.5 -25 12.5 curveto
-26.66 12.5 -28.25 11.84 -29.42 10.67 curveto
-31.25 4.59 -30.59 3.00 -29.42 1.83 curveto
-28.25 0.66 -26.66 0 -25 0 curveto
-23.34 0 -21.75 0.66 -20.58 1.83 curveto
-19.41 3.00 -18.75 4.59 -18.75 6.25 curveto
closepath fill showpage

Remark. Note the redundancies, for example newpath.

12.3 Escher’s sun and moon
I ‘scanned’ the picture and created appropriate paths in METAFONT by incorporating the points, specifying directions and using ordinary and splice joins. When finished METAFONT was ordered to write the curves – data for the spline pieces suitable for POSTSCRIPT – to the log file. This file was edited into a straight PostScript program. My poor man’s MFTOEPS, which is conceptually simple and does not require knowledge of clever tools.

The essential issues of the program are listed below.

path p[ ]; s=30;
p11=origin..(-.7s,.5s)..(-2s,0)..
(-2.6s,.3s)..(-3.5s,0);
p12=point 4 of p11..
38. Courtesy Jos Wijnink, who adapted my METAFONT program for the purpose.
Bijlage 5

{(1,3){-3.2s,s}{(3,1)}..(-2.2s,s)&
(-2.2s,1s){(-1,1)}..(-2.9s,1.9s)&
(-2.9s,1.9s){((1,1))..(right){(-1.5s,2.7s);}}..(-2.2s,s)

p13=point 6 of p12{(-1,-1)}..(-2s,2s)&
(-2s,2s){(-1s,2s)}..(-1.6s,1.6s)---
(-1.6s,1.6s){(-s,1s)}..(1,1)&
(1,1){(right){(-1s,2s)}..(-.6s,1.6s)&
(-.6s,1.6s){(-1s,2s)}..(-1.4s,1.4s)---
(-1.4s,1.4s){(-s,1s)}..(2,1.2s)&
(2,1.2s){(right){(-1s,2s)}..(-s,1.6s)}..(2,1.2s)point 0 of p11;

p1= reverse(p11..p12..p13..cycle);
fill p1;
%write to log file suitable for PS
"Bird1"

for k=0 upto 24:
show postcontrol k of p1;
show precontrol k+1 of p1;
show point k+1 of p1; "curveto";
endfor

%similar for other two (dark) birds
"spurious bird";

p4=point 0 of p33{(-2.2s,-2s)&
(-2.2s,-2s){(-1s,-2s)}..(-2s,-2s)&
(-2s,-2s){(-3s,-2s)}..(-2.5s,-2.5s)&
(-2.5s,-2.5s){(-3s,-2s)}..(-1.4s,-1.4s)&
(-1.4s,-1.4s){(-1s,-2s)}..(-2s,-2s)point 0 of p11;
draw p4;
%write to log file suitable for PS
%similar for Bird 5 and 6
showit; end

Note. "(string)" writes the (string) to the log file.
The PostScript program is a concatenation of curveto-s.

I guess the Sun and Moon picture could have been drawn easily by WYSIWYG graphics X-works software with a request for POSTSCRIPT output?

12.4 The squares patterns
Without explanation the codes for the simple squares patterns exercises are included. I hope it is clear enough. For the accurate calculation of the BB a little knowledge of go-niometry is needed.

%!PS-Adobe- Kepler’s squares, cgl Dec 96
%BoundingBox: -50 -37.5 50 50

\r 25 def %r dodecahedron
\s r 2.8284 75 sin mul div def
\d r 2 mul s .707 mul sub def
\rotsq(gsave s 0 moveto
 4(0 s lineto 90 rotate)repeat
 closepath stroke
 grestore)
}def
\dodeca{6(gsave r s sub 0 translate
  rotsq
  grestore 60 rotate
 )repeat
}def
\rstar{gsave 0 rstar translate dodeca grestore
  120 rotate
 )repeat
}\showpage

Interesting is the use of scaling in the code below.

%!PS-Adobe- More Square madness, cgl Feb 97
%BoundingBox: -80 -80 80 80
\r 50 def
\s .5858 r mul def /ms s neg def
\sds .707 s mul def /msds sds neg def
\element{gsave r star translate
 sds 0 moveto 0 sds lineto
 msds 0 lineto 0 msds lineto
 closepath fill
 grestore
}def

\f r sds sub r sds add div def
\gl 1 def
%
3{8(element 45 rotate)repeat
  f f scale /gl f gl mul def
  gl setgray
\gl setgray
}repeat \showpage

My case rest.

By the way, the right figure is classified by \{4, 6, 12\}, a nice layout for a herb garden. Have fun, and all the best.

Najaar 1997

67