Glisterings

Peter Wilson

Calm was the day and through the trembling air
Sweet-breathing Zephyrus did softly play—
A gentle spirit, that lightly did delay
Hot Titan's beams, which then did glister fair.

Prothalamion, Edmund Spenser

The aim of this column is to provide odd hints or small pieces of code that might help in solving a problem or two while hopefully not making things worse through any errors of mine.

Corrections, suggestions, and contributions will always be welcome.

This installment is not really about (LA)TEX, except peripherally.

Nothing in India is identifiable, the mere asking of a question causes it to disappear or to merge into something else.

A Passage to India, E. M. Forster

1 Reprise

Following the last column [4], Prof. Klaus Lagally wrote to me with another way of discarding an unwanted character at the end of a command. I had shown some code that acted in a similar manner to LATEX starred macros, except with a '?' instead of a '*'. The problem was to recognise the presence or absence of the character '?' and take different actions according to whether it was there or not, and to also discard the '?' if it was present. More precisely I presented

```
\makeatletter
\def\maybeQ{%
  \@ifnextchar ?{\@maybeQ}{\@maybe}}
\def\@maybeQ#1#2#3{Query (#2) and (#3).}
\def\@maybe#1#2{(#1) and (#2).}
\makeatother
```

Prof. Lagally instead suggested that `\@maybeQ` could be more simply defined as:

```
\makeatletter
\def\@maybeQ ?#1#2{Query (#1) and (#2).}
\makeatother
```

as a means of disposing of the '?'. In either version here are a couple of example results:

`\maybeQ{1st}{2nd} ->` (1st) and (2nd).
`\maybeQ?{1st}{2nd} ->` Query (1st) and (2nd).

Child! do not throw this book about!
Refrain from the unholy pleasure
Of cutting all the pictures out!
Preserve it as your chiefest treasure!

A Bad Child's Book of Beasts,
Hillaire Belloc

2 MetaPost and pdfLATEX

The MetaPost program generates PostScript illustrations. These can easily be inserted into a document to be processed by (LA)TEX to produce a `dvi` file. Generally speaking, though, pdfLATEX cannot handle PostScript files. Fortunately it can handle the limited form of PostScript that MetaPost generates, and so MetaPost illustrations can be directly embedded into a pdfLATEX document. This, though, is not quite as straightforward as it might be.

Given a file called, say, `figs.mp`, which contains perhaps three pictures, MetaPost will generate 3 files, `figs.1`, `figs.2` and `figs.3`, one for each picture. On the other hand, pdfLATEX expects MetaPost generated PostScript files to have an `.mps` extension. If you use the graphicx package you can get it to accept files with numeric extensions as though they had an `mps` extension by specifying:

```
\DeclareGraphicsRule{*}{mps}{*}{}
```

which tells `\includegraphics` to treat any extension it does not recognise as though it were `mps`.

LATEX, or at least programs like `dvips` or `xdvi`, can handle Encapsulated PostScript (`eps`) files, and you can perform similar magic for the graphicx package:

```
\usepackage{ifpdf}
\ifpdf
  \usepackage{graphicx}
  \DeclareGraphicsRule{*}{mps}{*}{}
\else
  \usepackage{graphicx}
  \DeclareGraphicsRule{*}{eps}{*}{}
\fi
```

If a MetaPost illustration might be used in a LATEX (as opposed to pdfLATEX) document, then put
`prologues := 1;`
at the start of the MetaPost file, which tells MetaPost to generate Encapsulated PostScript files. It seems to do no harm to use the same `prologues` specification for pdfLATEX.

A mathematician, like a painter or a poet, is a maker of patterns. If his patterns are more permanent than theirs, it is because they are made with ideas.

A Mathematician's Apology, G. H. Hardy

3 Spidrons

The other week I was idly glancing through *Science News* when I came across a short article about spidrons [3]; try googling for 'spidron' to get more on the subject. Spidrons, which were discovered and named by the Hungarian designer and graphic artist Dániel Erdély while doodling with hexagons, are made up of ever smaller connected triangles alternating between isosceles and equilateral in form.

It occurred to me that MetaPost could be used to draw these and after a little trial and error I came up with the following MetaPost program to support drawing spidrons.

```
%% semispid.mp  MP macro to draw a semi-spidron
% semispid(center, vertex, iterations,
%          color1, color2, clockwise)
def semispid(suffix $$, $)%
            (expr iter, shadea, shadeb, clock) =
if clock: hxa := -60; else: hxa := 60; fi
pair v[];
path phex[];
v0 := z$$;
v1 := z$;
% enclosing hexagon
for i := 2 upto 6:
  v[i] := v1 rotatedaround(v0,(i-1)*hxa);
endfor
z$a = v1; z$b = v2; z$c = v3;
z$d = v4; z$e = v5; z$f = v6;
phex0 := v1--v2--v3--v4--v5--v6--cycle;
if showverts:
  dotlabels.lft($a,$b,$c,$d,$e,$f);
fi
if showlines:
  draw v1--v3--v5--cycle;
  draw v2--v4--v6--cycle;
fi
% construct triangles
for n:= 1 upto iter:
  k := 10(n-1);
  j := 10n;
  v[1+j] := (v[1+k]--v[3+k])
            intersectionpoint
            (v[2+k]--v[6+k]);
  for i := (2+j) upto (6+j):
    v[i] := v[1+j]
            rotatedaround
           (v0, (i-1-j)*hxa);
  endfor
  if showlines:
    draw v[1+j]--v[3+j]--v[5+j]--cycle;
    draw v[2+j]--v[4+j]--v[6+j]--cycle;
  fi
  phex[n] := v[1+j]--v[1+k]--v[2+k]--cycle;
  phex[n+1] := v[1+j]--v[2+j]--v[2+k]--cycle;
  fill phex[n] withcolor shadea;
  fill phex[n+1] withcolor shadeb;
  if showcells:
    draw phex[n]; draw phex[n+1];
  fi
  if showedges:
    draw v[1+k]--v[1+j];
    draw v[2+k]--v[2+j];
  fi
endfor
if showedges: draw v[1+j]--v[2+j]; fi
if showhex: draw phex0; fi
enddef;
```

As its name implies, the routine `semispid` generates and draws half of a spidron, which Erdély called a semi-spidron, and this is contained within a hexagon. The location arguments are the center point of the enclosing hexagon and the location of one of the vertices. The other arguments control the number of triangles and two colors for coloring alternate triangles. The routine uses booleans, specified elsewhere, to control the display of various aspects of the construction method.

I used the next MetaPost program to create the spidron shown in Figure 1.

```
% glstr9.mp  MP spidron figures
prologues := 1;
input semispid
%%% define the boolean flags and defaults
% show the initial hexagon
boolean showhex; showhex := false;
% label vertices
boolean showverts; showverts := false;
% draw construction lines
boolean showlines; showlines := false;
% draw triangle cell boundaries
boolean showcells; showcells := false;
% work clockwise (yes = true)
boolean rh; rh := false;
% draw sem-spidron outline
boolean showedges; showedges := false;
% shading
color light,dark;
light := 0.1[white,black];
dark := 0.2[white,black];

beginfig(1); % a spidron
  u := 1in;  % units
showhex := false;
showverts := false;
showlines := false;
showcells := false;
rh := false;
showedges := false;
% center & initial vertex
z0 = (0,0);
z1 = (x0-2u,y0) rotatedaround(z0,60);
semispid(0, 1, 9, dark, light, rh);
y0-y1a = y1a-y10; x10=x0;
```

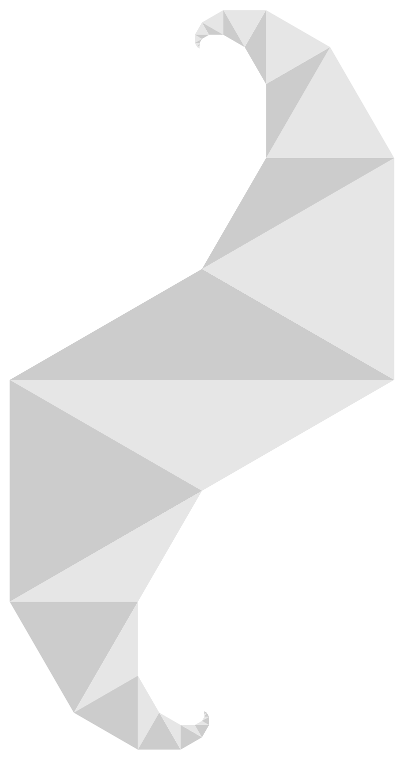

Figure 1: A spidron

```
z11 = z1b;
semispid(10, 11, 9, light, dark, rh);
endfig;
%% more pictures here
end
```

The construction details of a semi-spidron are illustrated in Figure 2. The `semispid` routine generates the vertices of a hexagon, labelling the given one as ‘a’, then the others in turn as ‘b’, ‘c’, etc. The hexagon is repeatedly partitioned by joining alternate vertices, which creates a smaller interior hexagon, which is then partitioned into a smaller one again, and so on until it all gets ‘too small’. The shaded triangles form a semi-spidron, starting on the ‘a-b’ side of the hexagon, and finishing close to the center. The second half of the complete spidron is a rotation of the first semi-spidron about the midpoint of the ‘a-b’ edge of the hexagon, with the colors reversed.

Spidrons are space-filling; that is, they can be assembled to completely cover, or tile, a plane surface. You can get a hint about this from Figure 3 which shows three semi-spidrons constructed in a single hexagon. The empty spaces can be exactly filled by three more semi-spidrons. A plane can be completely tiled using hexagons; in this particular case it happens that it can also be completely tiled by spidrons. Interesting effects can be achieved by changing the coloring of the spidrons. An example is shown in Figure 4. For much, much, more on tilings see *Tilings and Patterns* [2], although it doesn’t include spidrons as they hadn’t been discovered when the book was published.

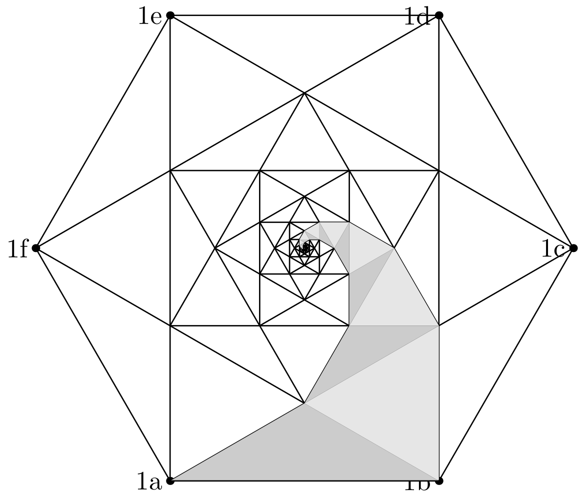

Figure 2: Construction details of a spidron

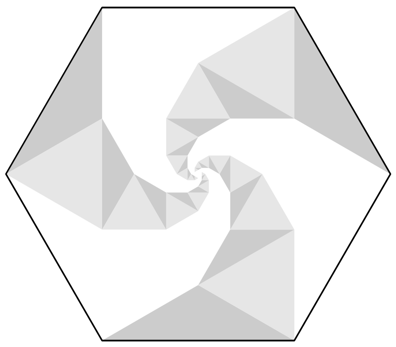

Figure 3: Three semi-spidrons in a hexagon

There is an associated figure that can also be made out of two semi-spidrons. In a spidron the two semi-spidrons are rotations of each other. In the shape that Erdély calls a *hornflake*, shown in Figure 5, the two halves are mirror images of each other. Unlike spidrons, hornflakes are not space-filling but can be used for tiling if they are suitably combined with spidrons, as can be seen in Figure 4.

In his article, Peterson says that

> [Erdély’s] insight was to start with an array of hexagons drawn on a sheet of paper and laid as if they were bathroom tiles. By creasing the pattern in the right combinations of mountains and valleys at the lines within each spidron arm and leaving a small

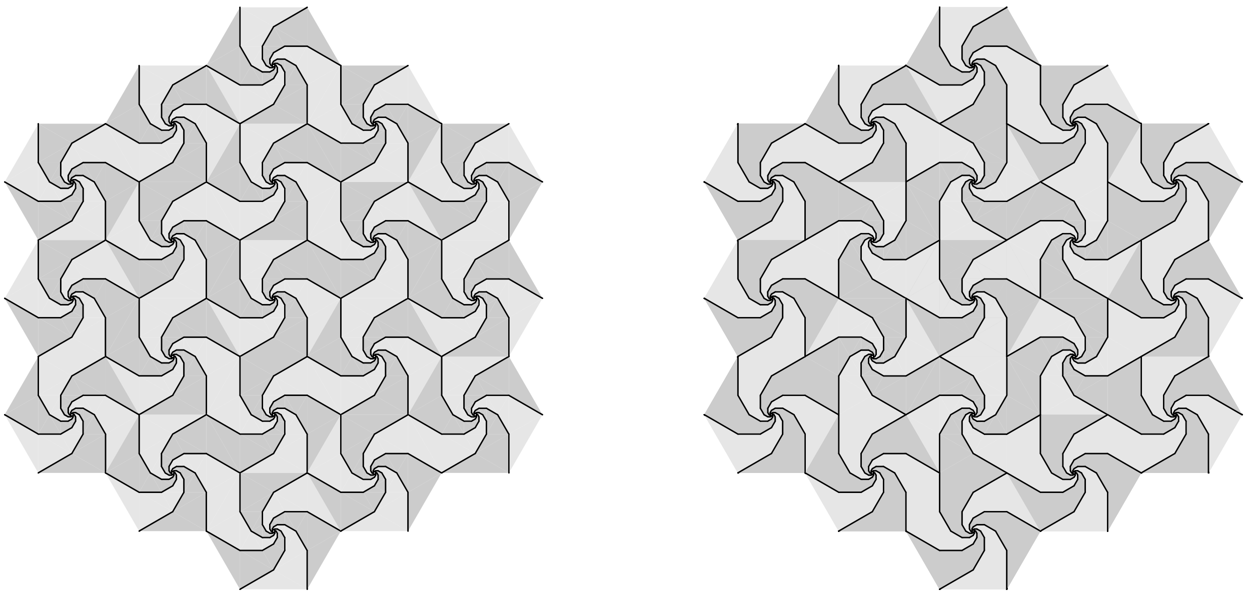

Figure 4: Tilings: (left) Spidrons can do it alone (right) Hornflakes need spidrons

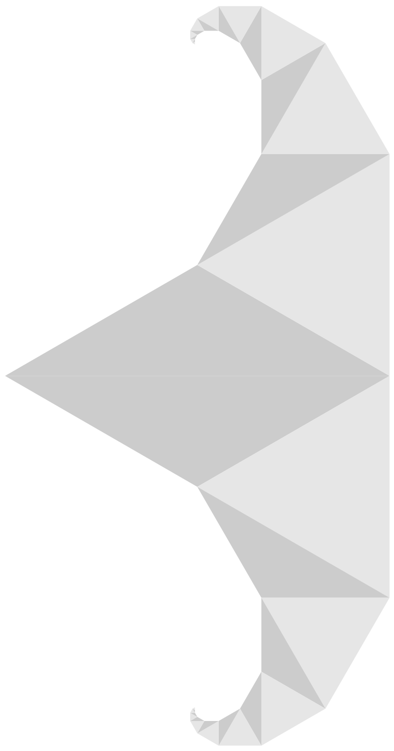

Figure 5: A hornflake

> hole at the center of each hexagon, he crinkled the whole aray into a dramatic three-dimensional relief.

It turns out that spidron patterns can also be assembled into novel three-dimensional crystal-like froms with spiral polygonal faces.

What is missing from the article is any hint as to what the 'right combinations' of folds might be to create these effects. After some searching on the web I found the following remarks by Erdély [1].

> I folded every second edge, reaching to the centre of the created hexagon in the given Spidron system, as a spine and folded every first edge as a groove. The resulting relief-like surface, under the impact of an external deforming force, does not show simple linear displacements, such as those produced with an accordion; instead, the edges between the vertices and the centres of the original hexagonal system move in a vortex within each hexagon.

After a lot of cogitation and physical experimentation I came to believe that among the 'right combinations' are the ones shown in Figure 6, which shows half a hexagon with three semi-spidrons. The dotted lines indicate 'valley' folds (paper on either side of the fold, or crease, is bent upwards) and the full lines indicate 'mountain' folds (paper on either side of the crease is bent downwards).

If you want to create a large construct for folding, here is the code for generating the spidron tiling

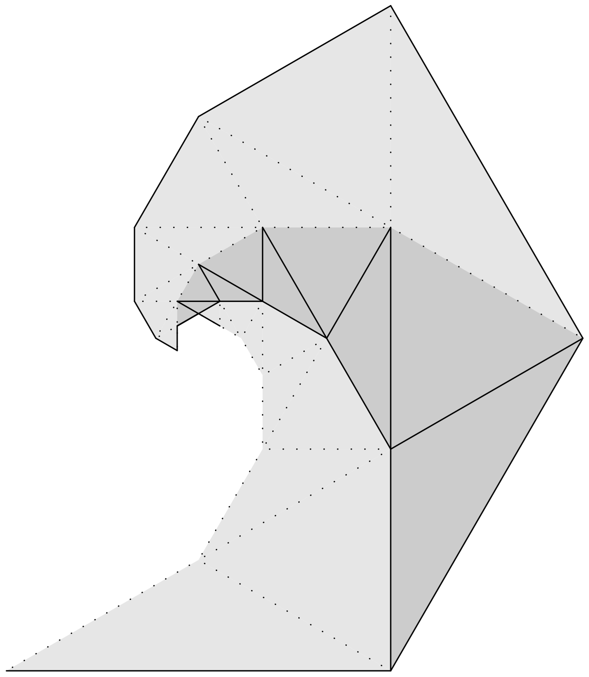

Figure 6: Folding

shown in Figure 4. You can, of course, modify this to meet your needs.

```
% glstr9.mp  MP spidron figures
% earlier pictures
beginfig(5); % spidron tiling
  u := 0.175in;
showhex := false;
showverts := false;
showlines := false;
showcells := false;
rh := false;
showedges := false;  showedges := true;
color cola, colb;
cola := light; colb := dark;
depth := 7;
rad := 2u;
z0 = (0,0);
% fill initial hexagon
for kn := 1 upto 6:
z[kn] = (x0-2u,y0) rotatedaround(z0,60*(kn-1));
if odd kn:
  cola := light;
else:
  cola := dark;
fi
 colb := cola;
semispid(0, [kn], depth, cola, colb, rh);
endfor
% copy (in circles) the filled hexagon
% to make the tiling
shd := (sqrt 3)/2*rad; % shift up/down
shr := 3rad;           % shift left/right
picture pic[];
pic100 := currentpicture;
pic0 := pic100 shifted (0,2shd);
for kn := 1 upto 6:
  pic[kn] := pic0 rotatedaround(z0,60kn);
  draw pic[kn];
endfor
pic10 = pic100 shifted (0,4shd);
for kn := 1 upto 6:
  pic[10+kn] := pic10 rotatedaround(z0,60kn);
  draw pic[10+kn];
endfor
pic20 = pic100 shifted (3rad, 0);
for kn := 1 upto 6:
  pic[20+kn] := pic20 rotatedaround(z0,60kn);
  draw pic[20+kn];
endfor
endfig;
% more pictures
end
```

However, I found that it was difficult enough to properly fold even a single large filled hexagon e.g., one that just fitted onto a typical sheet of paper, such as letter paper or A4. I decided that the best way was to use single spidrons, fold them appropriately, and then hinge them together with sticky tape. I then concluded that it was much more pleasurable to look at pictures of what others had accomplished (most of which, I suspect, were done using computer graphics instead of using physical methods and photographing the results).

References

[1] Dániel Erdély. Spidron system: A flexible space-filling structure. Idea 1979, first presented on the Twelfth International Conference on Crystal Growth in 1998, 2002. Possibly available at `http://www.szinhaz.hu/spidron`.

[2] Branko Grunbaum and G. C. Shephard. *Tilings and Patterns*. W. H. Freeman, 1987.

[3] Ivars Peterson. Swirling seas, crystal balls. *Science News*, 170(17):266–268, 21 October 2006.

[4] Peter Wilson. Glisterings. *TUGboat*, 29(2):324–327, 2008.

◇ Peter Wilson
18912 8th Ave. SW
Normandy Park, WA 98166
USA
`herries dot press (at) earthlink dot net`