Entry-level MetaPost 4: Artful lines

Mari Voipio

For basic information on running MetaPost, either standalone or within a ConTEXt document, see `http://tug.org/metapost/runningmp.html`. For previous installments of this tutorial series, see `http://tug.org/TUGboat/intromp`.

Thus far, we've done very little with lines except change their color. Other than that, we have used the built-in default settings for things like line width and line joins. In this tutorial we learn to tweak lines and use some special effects to put lines to work.

1 Line width

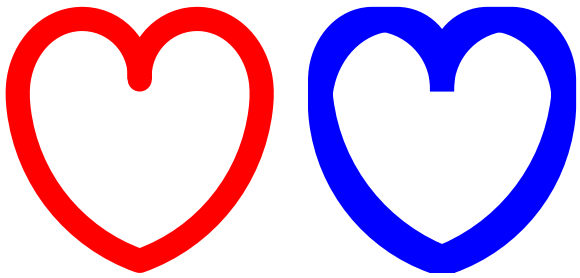
Lines are drawn with a pen. The default pen is `pencircle` with a line similar to what e.g. a ballpoint pen produces. When we want to change line width, we always need to specify both the pen "nib" and the width desired:

```
numeric u; u := 4mm; % measurement unit

path heart;
heart := (4u,0u) .. (0u,5u) .. (0u,6u)
  .. (2u,8u) .. (4u,6u) -- (4u,6u) .. (6u,8u)
  .. (8u,6u) .. (8u,5u) .. (4u,0u) -- cycle;

draw heart withpen pencircle scaled .8u
  withcolor red;
draw heart shifted (10u,0)
  withpen pensquare scaled .8u withcolor blue;
```

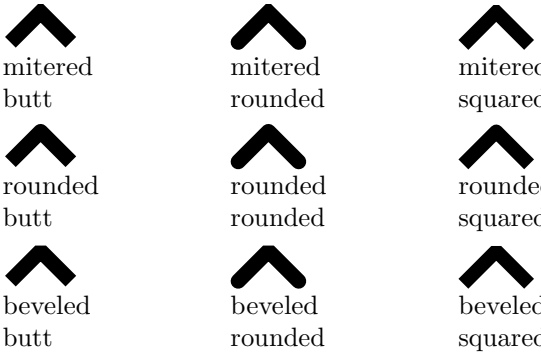
(Grayscaled for printed *TUGboat* output.)



For more information about pens, e.g. calligraphic pens, see the MetaFun manual [3, p. 36–38].

2 Line joins and line caps

Where lines are joined together, the corner can be turned in several different ways: "sharp" (`mitered`), rounded (`rounded`) or "cut off" (`beveled`). When a line is not cycled into a closed object, it also has two ends that can be "capped" in different ways: `butt` means straight end without any line cap, `rounded` adds a rounded line cap and `squared` adds a square cap to the end. If you look carefully at the example below, you can see that the butted line is shorter than the one with squared caps.



Linejoins and linecaps

How a linejoin looks depends also on the angle of the corner:

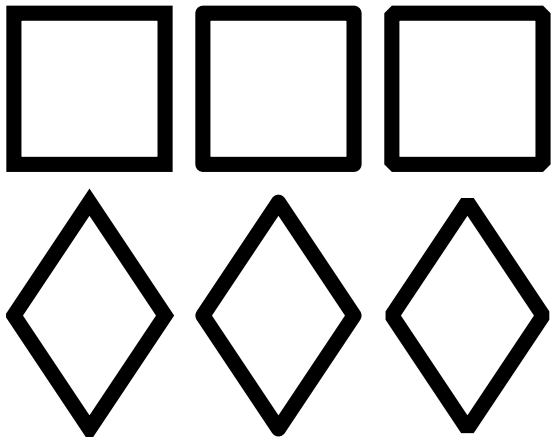
```
pickup pencircle scaled .2cm; % pen width

path rectangle; rectangle := (0,0) -- (2cm,0)
  -- (2cm,2cm) -- (0,2cm) -- cycle;
path diamond;   diamond := (1cm,0) -- (2cm,1.5cm)
  -- (1cm,3cm) -- (0,1.5cm) -- cycle;

linejoin := mitered;
draw rectangle shifted (0,3.5cm);
draw diamond;

linejoin := rounded;
draw rectangle shifted (2.5cm,3.5cm);
draw diamond shifted (2.5cm,0);

linejoin := beveled;
draw rectangle shifted (5cm,3.5cm);
draw diamond shifted (5cm,0);
```



3 Dashed lines

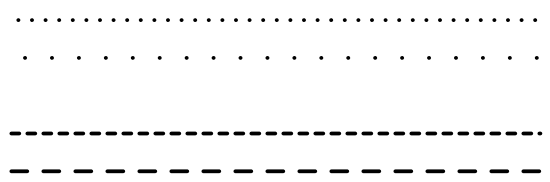
In addition to a solid line, we can create dotted and dashed lines. The distance between the dots/dashes is adjusted with the setting `scaled`; the bigger the

number, the more space there is between the dots or dashes.

```
pickup pencircle scaled .5mm; % pen width

% dotted lines
draw (0,20mm) -- (70mm,20mm) dashed withdots;
draw (0,15mm) -- (70mm,15mm) dashed withdots
  scaled 2;

% dashed lines
draw (0,5mm) -- (70mm,5mm) dashed evenly;
draw (0,0) -- (70mm,0) dashed evenly scaled 2;
```



It is not possible to fill paths that have a dashed (out)line. It is also advisable to use only `pencircle` in combination with dashed lines.

For more information on adjusting dashed lines, see the MetaPost manual [2, pp. 37–40] and the MetaFun manual [3, pp. 40–41].

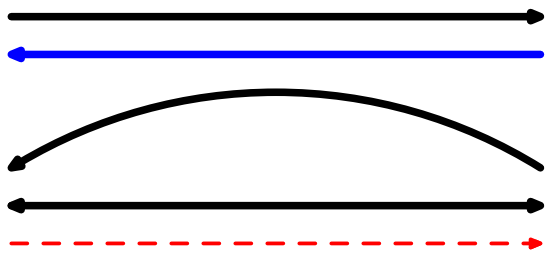
4 Arrows

There are separate commands for drawing arrows, but they have the same settings as the plain `draw` command: pen, dashing and color. Arrows go from left to right by default; an arrow with the arrowhead on the left can be created either by giving the coordinates right-to-left or by using the `drawarrow reverse()` command.

```
pickup pencircle scaled .1cm;

drawarrow (0,3.5cm) -- (7cm,3.5cm);
drawarrow (7cm,3cm) -- (0,3cm) withcolor blue;
drawarrow reverse((0,1.5cm) .. (3.5cm,2.5cm)
 .. (7cm,1.5cm));
drawdblarrow (0,1cm) -- (7cm,1cm);

drawarrow (0,0.5cm) -- (7cm,0.5cm)
  withpen pencircle scaled .05cm
  dashed evenly scaled 2
  withcolor red;
```



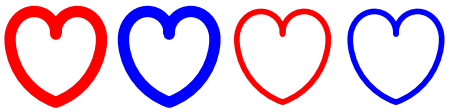
5 Applying settings for multiple paths

In the examples above we have already used `pickup` to set the pen width for multiple paths. I think of this as having a bunch of pens on the table and picking up one after another to draw with; the drawn lines have the same width until I switch to a different pen — except that we can override the pickup settings for an individual path by using the `withpen` command, as in the arrow example above. The following example picks up two pens in turn:

```
numeric u; u := 1.5mm;
heart := (4u,0u) .. (0u,5u) .. (0u,6u)
  .. (2u,8u) .. (4u,6u) -- (4u,6u) .. (6u,8u)
  .. (8u,6u) .. (8u,5u) .. (4u,0u) -- cycle;

pickup pencircle scaled u;
draw heart withcolor red;
draw heart shifted (10u,0) withcolor blue;

pickup pencircle scaled .5u;
draw heart shifted (20u,0) withcolor red;
draw heart shifted (30u,0) withcolor blue;
```



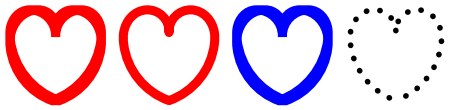
A more versatile command is `drawoptions(`*pen, color, withcolor*) which allows us to set default line properties: pen `withpen`), color (`withcolor`) and dashing (`dashed`). This command can be used to set defaults for the whole drawing or just part of it, and is valid until the next `drawoptions()` command. You can also reset everything by giving the `drawoptions` command with nothing within the parentheses.

The `drawoptions` are overridden by setting pen and/or color and/or dashing individually for a path, as usual:

```
numeric u; u := 1.5mm;
heart := (4u,0u) .. (0u,5u) .. (0u,6u)
  .. (2u,8u) .. (4u,6u) -- (4u,6u) .. (6u,8u)
  .. (8u,6u) .. (8u,5u) .. (4u,0u) -- cycle;

drawoptions(withpen pensquare scaled .8u
            withcolor red);

draw heart; % default settings from drawoptions
draw heart shifted (10u,0)
  withpen pencircle scaled .8u; % pen override
draw heart shifted (20u,0)
  withcolor blue; % color override
draw heart shifted (30u,0)
  dashed withdots
  withpen pencircle scaled .5u
  withcolor black; % dash, pen, color override
```



6 MetaFun bonus: Grids

With the MetaFun package we can easily create evenly spaced and logarithmic grids. The syntax is:

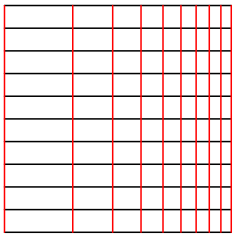
```
% horizontal/vertical linear:
hlingrid (Min, Max, Step, Length, Width)
vlingrid (Min, Max, Step, Length, Height)

% horizontal/vertical logarithmic:
hloggrid (Min, Max, Step, Length, Width)
vloggrid (Min, Max, Step, Length, Height)
```

The grid settings are used in combination with the `draw` command:

```
pickup pencircle scaled .2mm;

draw hlingrid (0, 10, 1, 3cm, 3cm);
draw vloggrid (0, 10, 1, 3cm, 3cm)
  withcolor red;
```



We can create gridded paper with the right grid settings:

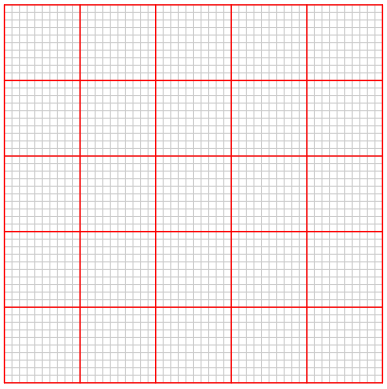
```
width  := 5cm; height := 5cm; unit := cm;

drawoptions(withpen pencircle scaled .2pt
            withcolor .8white);
draw vlingrid(0, width /unit, 1/10, width,  height);
draw hlingrid(0, height/unit, 1/10, height, width );

drawoptions(withpen pencircle scaled .5pt
            withcolor red);
draw vlingrid(0, width /unit, 1,    width,  height);
draw hlingrid(0, height/unit, 1,    height, width );
```

We haven't seen the `unit` assignment (in the first line) before: in MetaFun, it applies to numbers without any other unit.

See the MetaFun manual [3, pp. 213–215] for further examples.



7 References

[1] Running MetaPost and Metafun.
`http://tug.org/metapost/runningmp.html`

[2] MetaPost manual.
`http://tug.org/docs/metapost/mpman.pdf`

[3] MetaFun manual.
`http://www.pragma-ade.com/general/manuals/metafun-p.pdf`

◇ Mari Voipio
`mari dot voipio (at) lucet dot fi`
`http://www.lucet.fi`