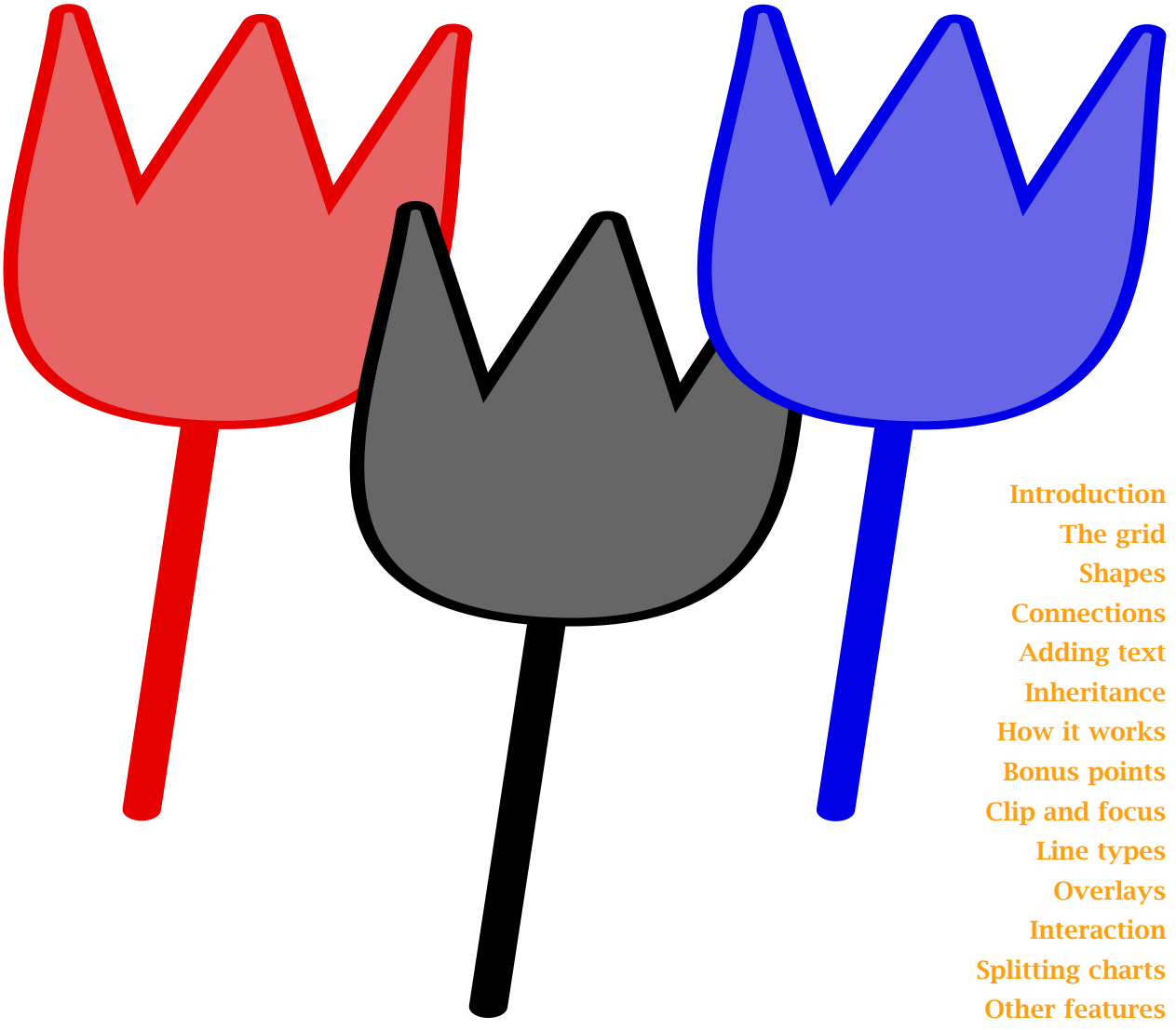
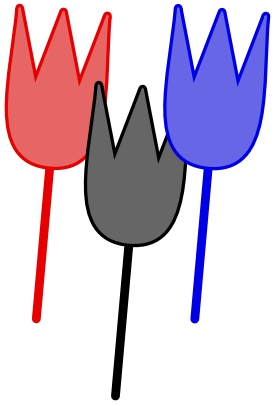
ConTeXt
up-to-date
1999/3

Flowcharts



Introduction
The grid
Shapes
Connections
Adding text
Inheritance
How it works
Bonus points
Clip and focus
Line types
Overlays
Interaction
Splitting charts
Other features

Pragma ADE
Ridderstraat 27
8061GH Hasselt NL



Introduction

This is just another story of TEX meeting METAPOST. This time we will focus on charts, especially flowcharts. In CONTEXT flowchart support is not part of the core functionality, but, at least for the moment, presented in a module. Therefore, before one can actually define a chart, this module must be loaded:

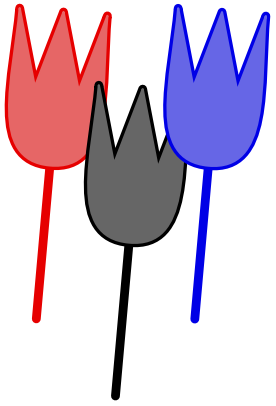
```
\usemodule[chart]
```

Once loaded, the user has access to the functionality described here. Before we go into detail on the features, we will say some words on history.

When dealing with graphics, it makes sense to use a drawing program. In fact, before we started using this module, we did use such programs, and they have without doubt their advantages. As soon as CONTEXT supported interactive documents, there were means to make graphics interactive, and as long as only a few graphics are involved, this mechanism works ok.

And then we suddenly had to make a document with thousands of pages and hundreds of often rather complicated flowcharts. Because these charts were tightly integrated in the main document, they not only had to be consistent in the use of fonts, but also had to be interactive and were to be presented both as a whole and in subchart parts. We wanted fonts, colors and the overall appearance as well as names of people, places, steps, activities and more to be consistent, especially because these charts are constantly updated.

I use the term flowchart here because I want to stress that this module typesets charts which cells are connected by lines. Our first application of this module concerned diagrams that expressed actions and relations between those actions, using some techniques originating years ago in programming environments: lines were not to cross, one should read from top to bottom and left to right, etc. However, the module presented here can be used to draw all kind of charts, and all kind of connections.



The grid

A flowchart consists of shapes, positioned on a grid, connected by lines. The grid enables the user to anchor the shapes and enables the drawing routines to determine connections. One can either explicitly specify the grid, or let it be calculated automatically.

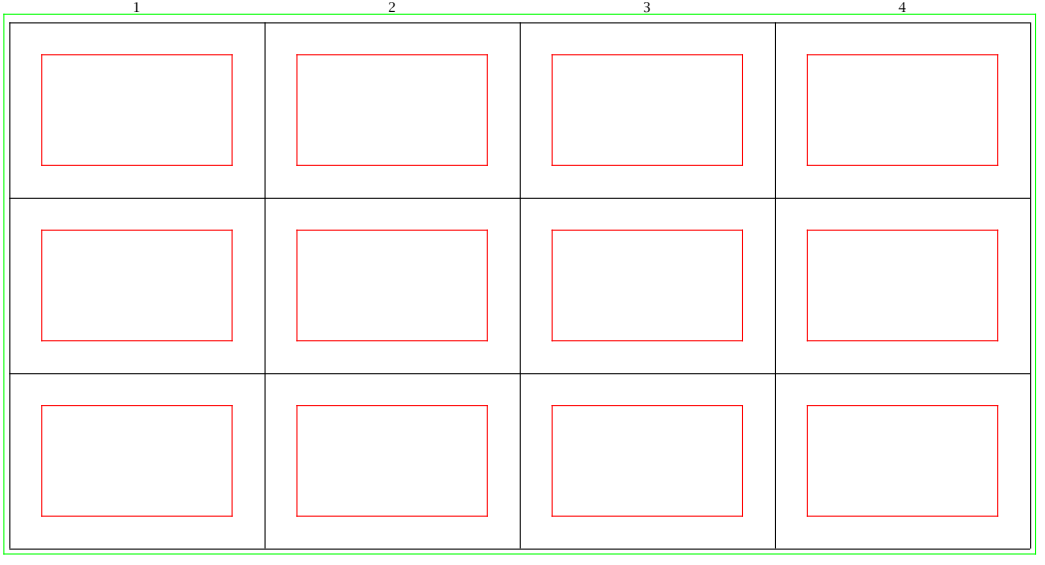
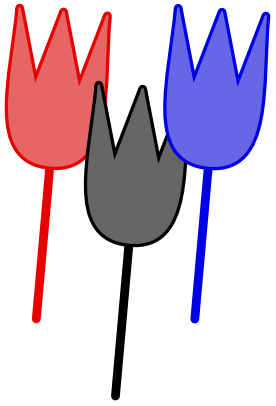


Figure 1 The grid.

Normally the grid is not visible, unless one enters test mode. The grid in figure 1 is the result of the definition:

```
\setupFLOWcharts
  [nx=4,
   ny=3,
```



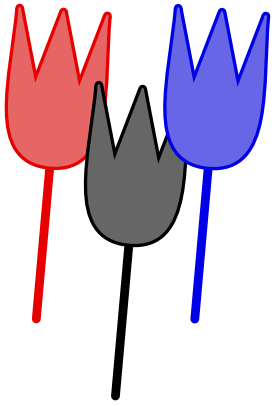
```
    dx=2\bodyfontsize,
    dy=2\bodyfontsize,
    width=12\bodyfontsize,
    height=7\bodyfontsize,
    maxwidth=\textwidth]

  \startFLOWchart [grid]
  \stopFLOWchart
```

The most straightforward way of calling up this chart is by saying:

```
  \FLOWchart[grid]
```

In figure 1 the inner rectangles represent the average size of the shapes. The lines around these shapes represent the grid cells. The outer rectangle shows the offset. Shapes are smaller than grid cells. This is necessary because connecting lines need some room. The offset is important, because when a connection follows the outer lines, a little extra space outside that line not only looks better, but also prevents the line from being clipped. It makes sense to keep the offset as well as the space between shapes constant across a document. The numbers are typeset outside the bounding box of the figure.

Grid cells are numbered from top to bottom starting at the left side, so the left topmost cell is $(1,1)$. Later we will see that because cells have names, these numbers play a minor role.



Shapes

A shape is something, typically a text, within a frame. The frame has certain dimensions and can have some color and background. In this respect it looks like the ConTeXt command `\framed`. The most important shapes have been assigned names as indicated in figure 2. There are more shapes, but they are identified by a number only. The total number of shapes will quite certainly increase. The shapes *up*, *down*, *left* and *right* are not really shapes, but lines that can be used to force a direction.

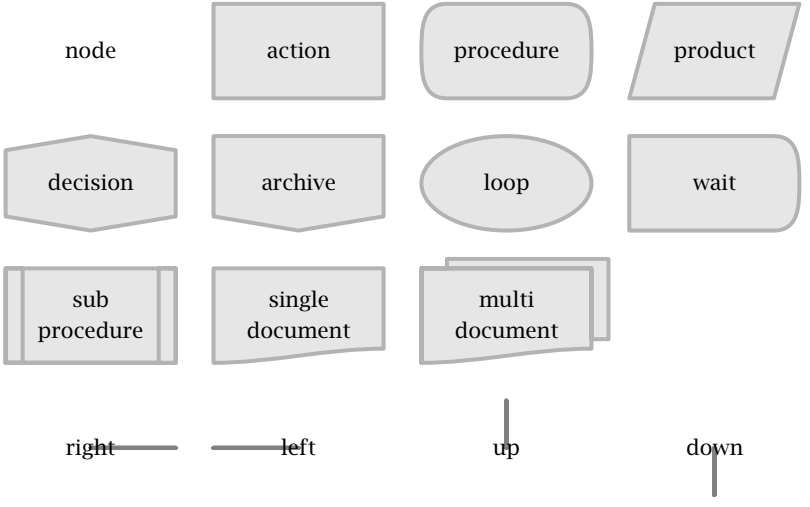
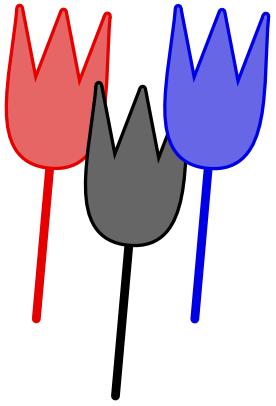


Figure 2 The shapes.

When no shape is specified, the default shape is used. One can change this default value with the `\setupFLOWshapes` command.



```
\startFLOWchart [cells]
  \startFLOWcell
    \name     {first}
    \location {1,1}
    \shape    {singledocument}
    \text     {not realy a document}
  \stopFLOWcell
\stopFLOWchart
```

A flow chart consists of cells. Each cell has a name, is positioned somewhere on the grid, has a certain shape, and normally this shape surrounds text. The shape is drawn by METAPOST, and the text is placed by TEX. Later we will see that there are some more fields to fill. Names are local to a chart.

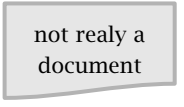
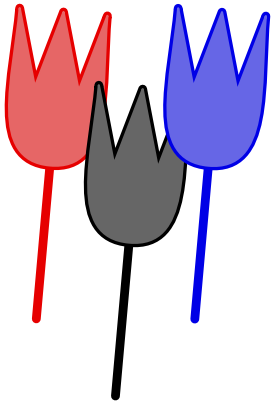


Figure 3



Connections

Shapes can be connected. As shown in figure 4 each shape has four connection points: top, bottom, left and right. When connecting shapes we refer to their logical names and specify two of the four directions.

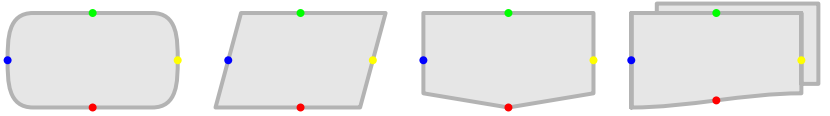


Figure 4 The connection points.

In figure 5 we see three connections. The lines have smooth curves and run across the grid lines. By using smooth curves, an option that can be turned off, the direction of touching curves is always clear. Here we use arrows. Smoothing, arrows and dashed lines are some of the attributes of lines.

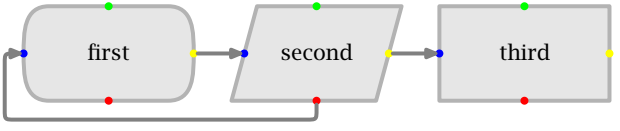
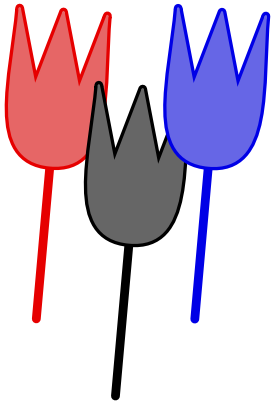


Figure 5 A few connections.

There can be more than one connection per shape. When defining such a connection we first specify the direction. In this example `[rl]` means connect the right point to the left one, while `[tr]` results in a connection between the top and the right point. The second argument specifies the shape to connect to. As we can see, connections can point back to their origin shape.



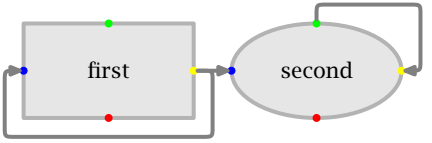


Figure 6 A few more connections.

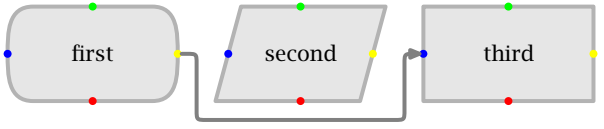
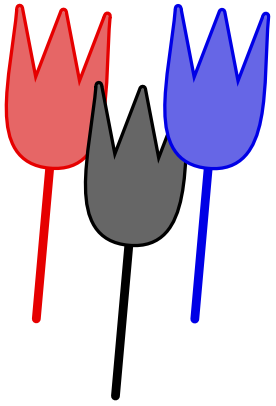


Figure 7 Going around shapes.

The connection drawing routines have a rather strong urge to follow grid lines. Figure 8 demonstrates this several times. From the first shape to the third one, we see that the connection takes the shortest route possible without crossing other shapes. I have to admit that the routines in themselves are rather stupid, but for normal use they suffice.

Generally speaking, when two lines end at the same point, it makes sense to connect these. When on the other hand lines originate at the same point or cross eachother, readers can get confused. Therefore such lines are drawn in such a way that they don't touch. In this respect, figure 9 demonstrates a less than optimal chart.



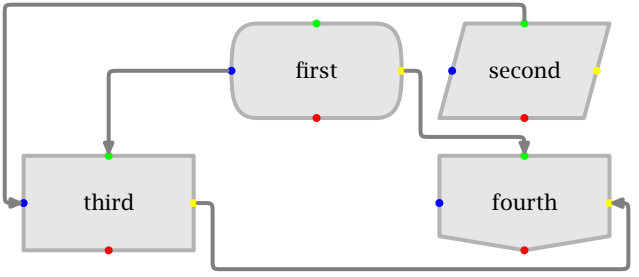


Figure 8 Following grid lines.

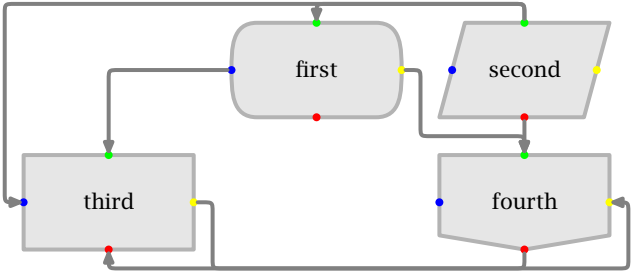
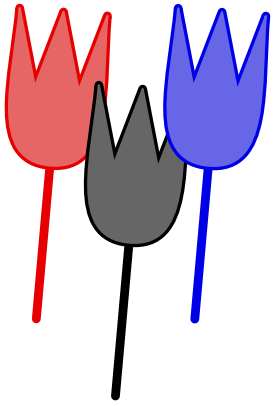


Figure 9 Confusing (crossing) grid lines.



Adding text

In figure 10 we have added some comment to a connection. Like the dots at the connections, the point halfway the connection shows up in a special debugging mode. The comment will be placed relative to this point. In figure 10 this is to the left of the point.

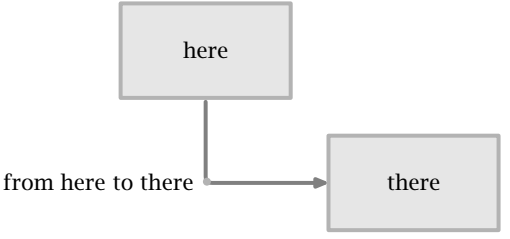
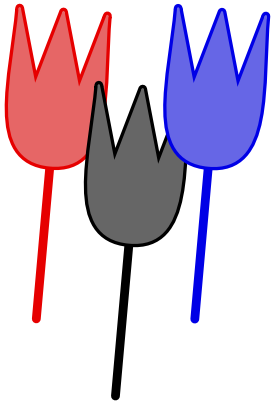


Figure 10 Comment to connections.

It will be no surprise that a comment is defined using `\comment`. Comments can be anchored to eight locations, simply `l`, `r`, `t`, `b`, or a combination like `tr`.

```
\startFLOWcell
  ...
  \comment [l] {from here to there}
  ...
\stopFLOWcell
```

We can also put labels at the connection points. Often this is preferred over comment halfway along a connection. Like comments, labels have a dedicated command. Here we specify the connection point `l`, `r`, `t` or `b`.



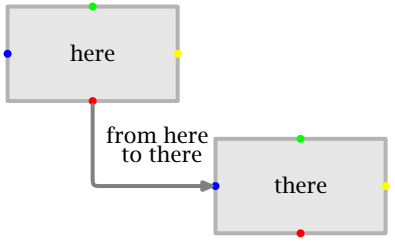


Figure 11 Labels to connection points.

```
\startFLOWcell
  ...
  \label [1] {to there}
  ...
\stopFLOWcell
```

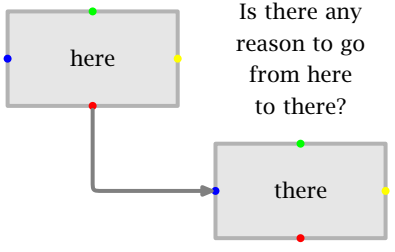
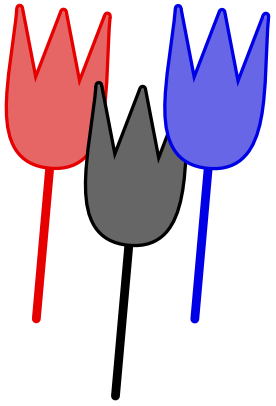


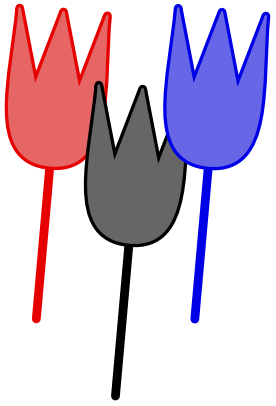
Figure 12 Text without shapes.

In figure 12 we see some text without any shapes around it. When shape `none` is specified, the whole shape area is available for text.



```
\startFLOWcell
  \shape    {none}
  \location {2,1}
  \text     {Is there any reason to go
             from here to there?}
\stopFLOWcell
```

One can force the alignment with the key characters `l`, `r`, `c`, `t` and `b`. So, the next definition only places text.



Inheritance

When explaining something by using a chart, we often show successive versions of the chart, where each version adds a new feature. To prevent us from retyping the same components again and again, it helps to partition the source code of the complete chart into subcharts. Inclusion of a part is straightforward: the subchart is called by name and positioned on the grid.

```
\startFLOWchart [include]
  \includeFLOWchart[labels][x=1,y=1]
  \startFLOWcell
    \shape    {none}
    \location {2,1}
    \text     {There is no reason to go
               from here to there!}
  \stopFLOWcell
\stopFLOWchart
```

The included sub chart has its own reference point, so one does not have to bother about positions.

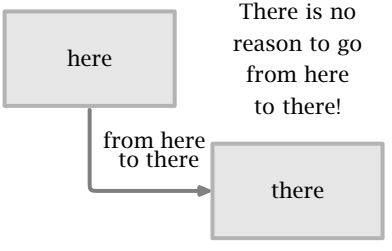
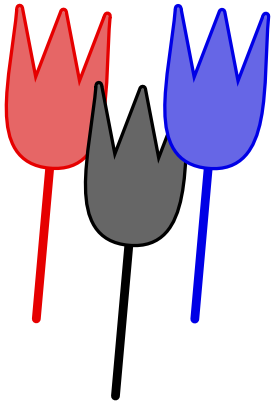


Figure 13 Sharing components.



How it works

The charting module, loaded by `\usemodule[chart]` is only responsible for the TEX part of the job, which means positioning text and graphics generated by METAPOST. The grid, shape and connection drawing routines are grouped together in a dedicated METAPOST module.

Because of the mix of TEX and METAPOST, and because we want to be able to scale charts, the buffer mechanism is used. The communication between TEX and METAPOST uses the METAPOST embedding macros that are native to CONTEXT. Additional communication from METAPOST to CONTEXT is handled in the module itself. This rather fuzzy description is visualized in figure 14. Depending on the general color settings, among the other processes involved are: color conversion and/or reduction to grayscales, and conversion to PDF. When watching this module in action, don't feel disturbed by the many steps involved.

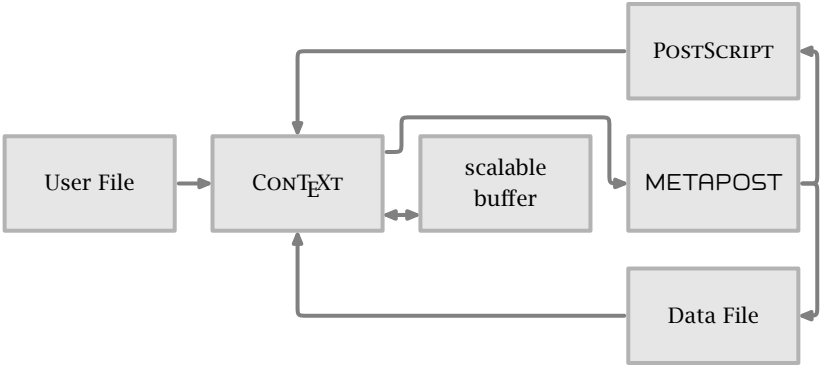
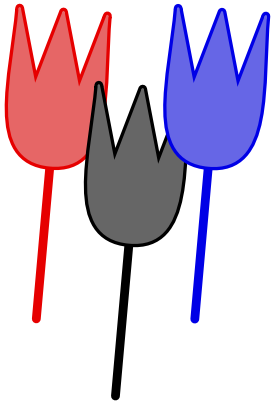


Figure 14 The process.



A few pages back we introduced the named shapes. There are however some more shapes. Each shape is identified by a number. In figure 15 all currently available shapes are shown. The zero numbered shape is actually a shape, but with zero dimensions. It can be used as a meeting point for lines. The unused numbers can be used for new shapes. The maximum number of shapes is limited to 4095, which is a METAPOST limitation.

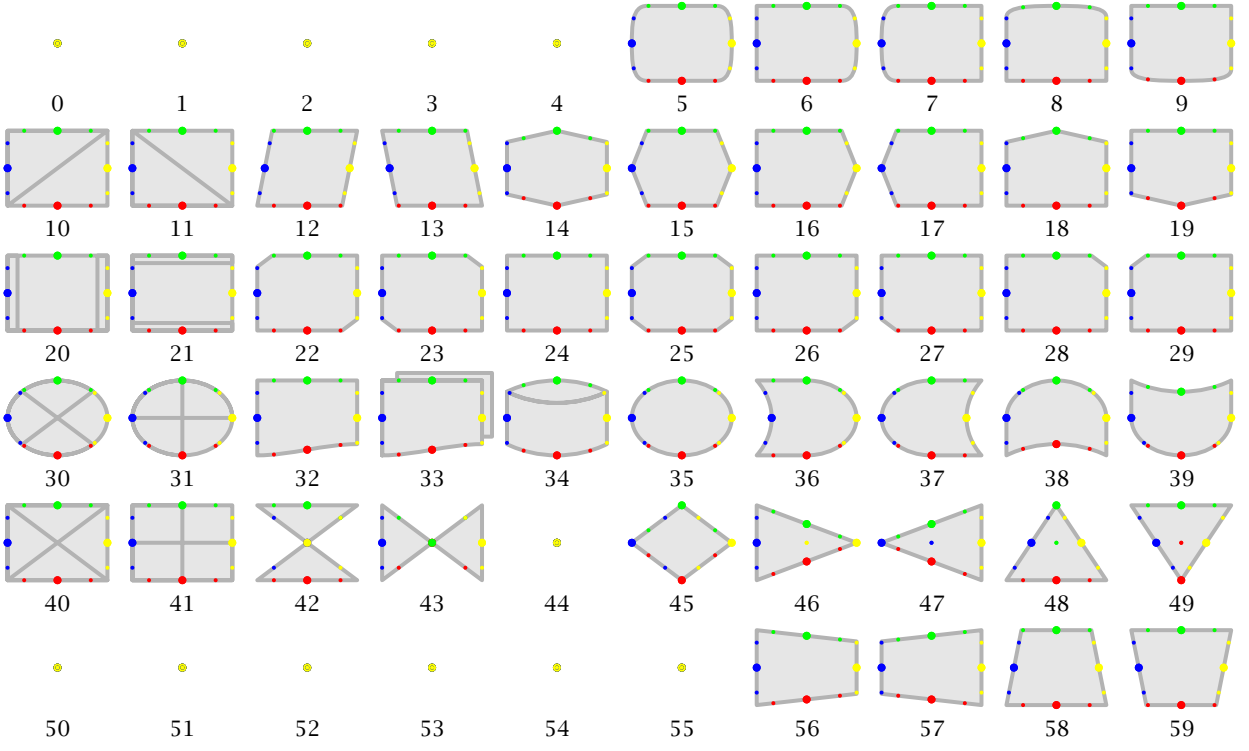
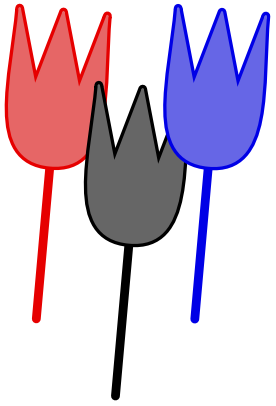


Figure 15 All shapes by number.



Bonus points

Sometimes charts can become rather large, due to the many shapes used or lines drawn. At the same time charts can become unclear because more than one connection starts or ends at a shape. Figure 16 shows a way out of this situation.

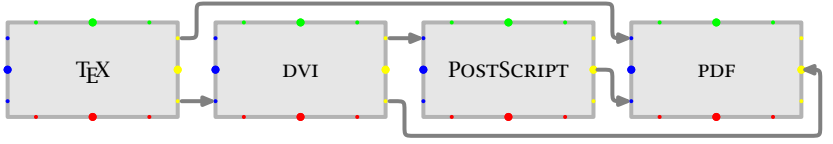


Figure 16 Even more points.

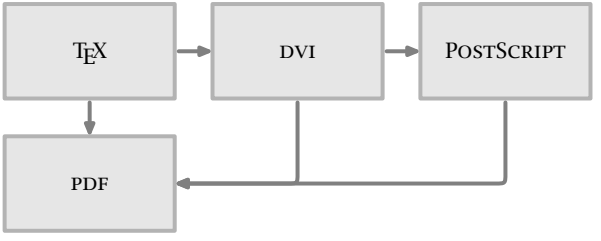
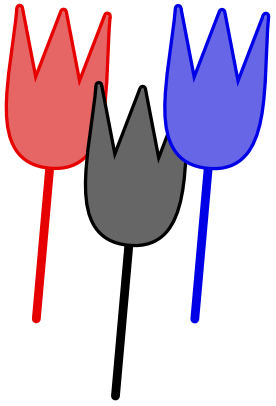


Figure 17 An alternative for figure 16.

Defining such a chart is not so much harder than previous cases. Each left, right, top and bottom point has two companion points: positive and negative. In connections the left points are: `pl`, `l` and `nl`: positive left, left and negative left, so the first cell in figure 16 is defined as:

```
\startFLOWcell
  \name       {tex}
```



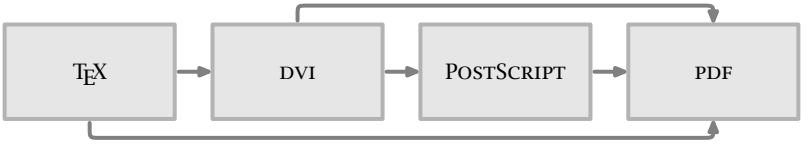
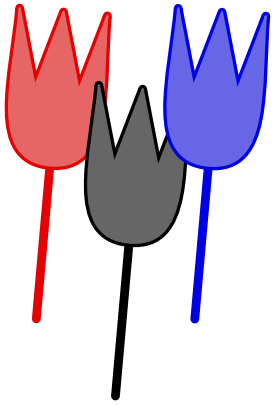


Figure 18 Yet another alternative for figure 16.

```
    \location   {1,1}
    \shape      {action}
    \text       {\TEX}
    \connection [prpl] {pdf}
    \connection [nrnl] {dvi}
  \stopFLOWcell
```

Alternatively to p and n one may use + and -. As soon as the positive and negative points are used, the connection drawing routines will take into account the fact that they are off-center. This does not free users from thinking about better ways to draw such a chart.



Clip and focus

The flowcharter automatically calculates the size of the grid. When needed, one can force the dimensions and/or clip pieces of a chart. Figure 19 shows such a clip. This example also shows why the offset, the small area around the outer grid lines, is important. Figure 19 was produced while the next settings were in action.

```
\setupFLOWcharts
  [x=1,y=1,nx=2,ny=1]
```

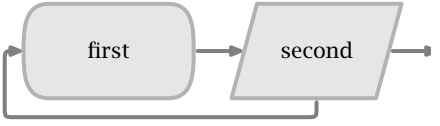


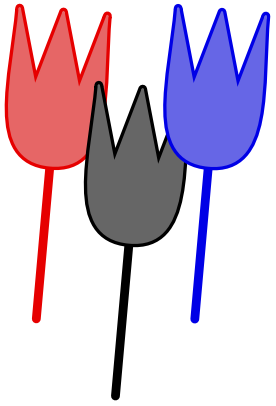
Figure 19 Clipping a piece of a chart.

Sometimes, for instance when explaining a chart, it makes sense to emphasize one or more particular cells. Therefore this module offers the ability to focus on cells. In figure 20 we see that the focus is on the cell with name `dvi`. This is accomplished by saying:

```
\setupFLOWfocus
  [framecolor=pragmacolor]
\setupFLOWcharts
  [focus=dvi]
```

Clipping and focus bring us to the third way of zooming in: autofocus. Figure 21 shows what happens when we say:

```
\setupFLOWfocus
  [framecolor=pragmacolor]
```



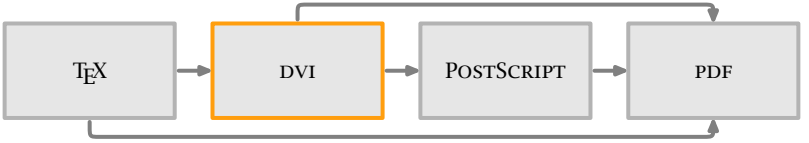


Figure 20 Gaining some focus.

```
\setupFLOWcharts
  [focus=dvi,autofocus=dvi,
   nx=1,ny=1]
```

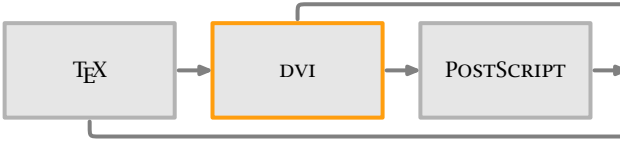
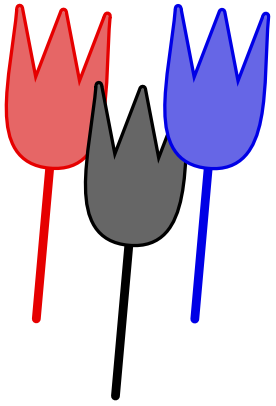


Figure 21 Applying autofocus.

In figure 21 we see both focus and auto focus in action.



Line types

As is to be expected, we can set up some characteristics of a chart, the shapes, the connecting lines, and the focus. When we want dashed lines and a different shape color, we just say:

```
\setupFLOWshapes
  [framecolor=pragmacolor]
\setupFLOWlines
  [framecolor=pragmacolor,
   dash=yes]
```

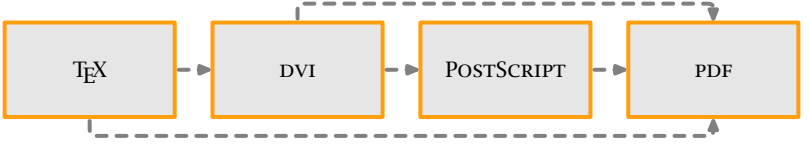
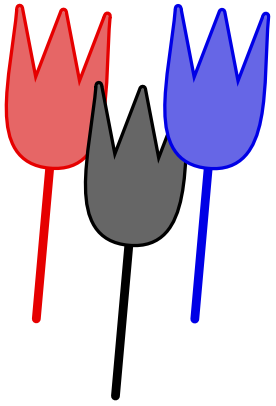


Figure 22 Dashed and colored lines.

We can change the characteristics at several levels: chart, line, shape and/or focus. In the near future some more options will be added. Once the METAPOST module is stable and documented, the graphic code will also be accessible. The formal definition of the four setup commands is as follows (these diagrams conform the CONTEXT conventions):



\setupFLOWcharts[..,..=..,..]

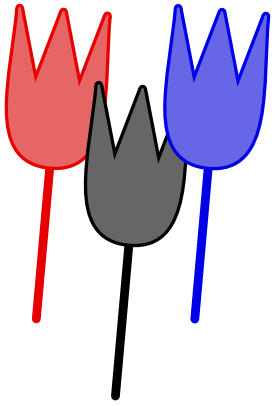
option	test
width	*dimension*
height	*dimension*
offset	*dimension*
dx	*dimension*
dy	*dimension*
nx	*number*
ny	*number*
x	*number*
y	*number*
autofocus	*name*
focus	*name*
backgroundcolor	*name* white

\setupFLOWfocus[..,..=..,..]

..=.. see \setupFLOWshapes

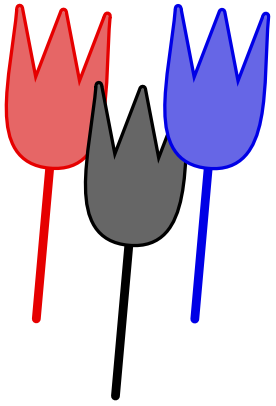
\setupFLOWlines[..,..=..,..]

corner	round rectangular
arrow	yes no
dash	yes no
radius	*dimension*
color	*name* FLOWlinecolor
rulethickness	*dimension*
offset	overlay none



```
\setupFLOWshapes[..,..=..,..]
```

framecolor	*name* FLOWframecolor
default	*name* action
background	color screen
backgroundcolor	*name*
backgroundscreen	*number*
rulethickness	*dimension*
offset	overlay none *dimension*



Overlays

Why should we limit ourselves to text? In ConTeXt most frames–related features can have overlays. Because in this flowchart module shapes are drawn by MetaPost, we use a slightly different approach: there can be overlays but they are sort of clipped by the shape. Figure 23 illustrates this.

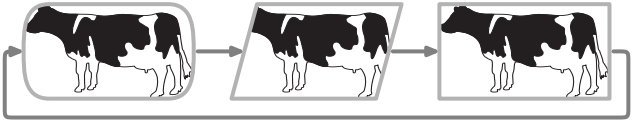


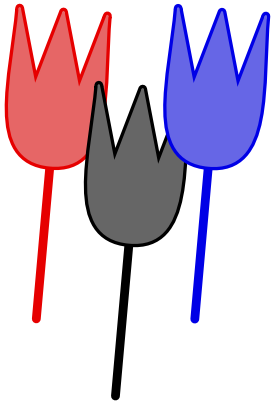
Figure 23 Overlays.

There are two commands related to overlays. In our example we used:

```
\startFLOWcell
  ...
  \figure {cow}
  ...
\stopFLOWcell
```

which automatically scales figure *cow* to the shape size. An alternative command, that offers a bit more control, is:

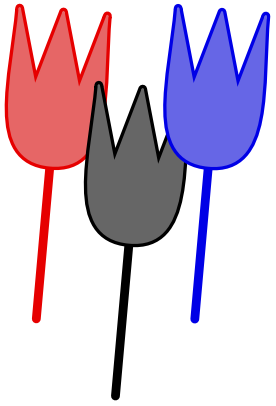
```
\startFLOWcell
  ...
  \overlay {coward}
  ...
\stopFLOWcell
```



Here we call for an overlay, for instance defined by

```
\defineoverlay
  [coward]
  [{\externalfigure
      [cow]
      [width=\overlaywidth,
       height=\overlayheight]}]
```

The normal rules for overlays apply. The width and height of the overlay are available at the moment the overlay is typeset.



Interaction

One of the reasons for writing this module was that we wanted interactive flowcharts. The shape text can contain references.[1] The shape as a whole becomes a button as soon as we add the `\destination` entry:

```
\startFLOWcell
  ...
  \destination {destination}
  ...
\stopFLOWcell
```

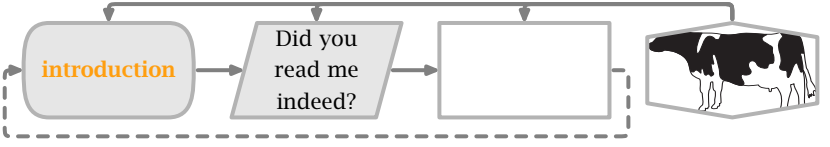
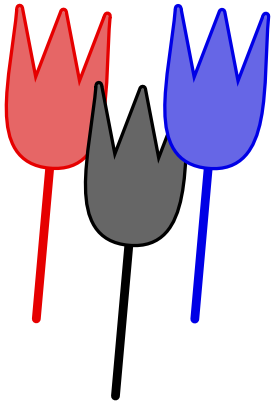


Figure 24 Overlays, help and buttons.

When read as PDF file, figure 24 shows quite a few interactive features. The first cell has a `\goto` included in the text, like:

```
\startFLOWcell
  ...
  \text {\goto{introduction}[introduction]}
  ...
\stopFLOWcell
```

[1] Currently this only works ok when the chart is not scaled.



The second shape is interactive as a whole. This is accomplished by saying:

```
\startFLOWcell
  ...
  \text        {Did you\\read me\\indeed?}
  \destination {introduction}
  ...
\stopFLOWcell
```

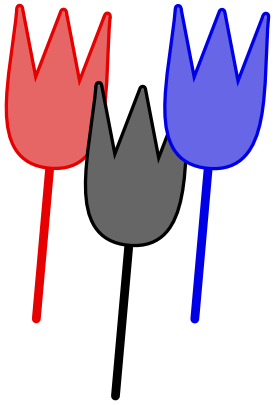
The third cell is a movie, which is only visible in suitable viewers. The movie is included as an overlay.

```
\defineoverlay
  [fun]
  [{\externalfigure
      [texwork.mov]
      [width=\framedwidth,
       height=\framedheight,
       preview=yes,
       repeat=yes]}]
```

The last cell is an illustration. Apart from the third cell, each cell has additional help information attached. Help information is defined with:

```
\startFLOWcell
  ...
  \help {identifier}
  ...
\stopFLOWcell
```

and defined by:

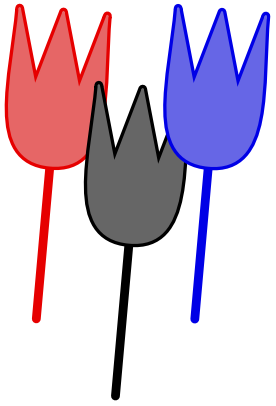


```
\starthelptext [identifier]
  ... text ...
\stophelptext
```

Helpinfo is placed by `\helpdata` and removed by executing the reference `HideHelp`. The macro returns a centered set of hidden help texts, that can be placed somewhere by the normal layout commands. The help information mechanism is independent from the flowchart module and keeps track of the pages where help is available.

The help information pops up when one points and clicks below a cell. By putting the help button there, it has less chance to interfere with other interactive things. Of course users should be made aware of this feature.

Summarizing, we can put text in a shape, provide one or more overlays to this shape, make cells and/or part of the text in a shape into a hyperlink, and show help when requested.



Splitting charts

Sometimes a chart does not fit comfortably on the page. In such cases, it makes sense to split up the chart. There is a dedicated setup command to serve splitting:

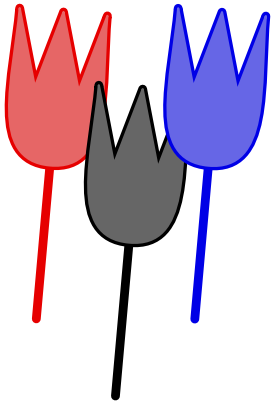
```
\setupFLOWsplit[..,..=..,..]
nx          number
ny          number
dx          number
dy          number
command     command
before      command
after       command
marking     on off
```

An example of splitting is:

```
\setupFLOWsplit
  [nx=5,ny=10,
   dx=1,dy=1,
   before=,after=\page]

\FLOWcharts[mybigflow]
```

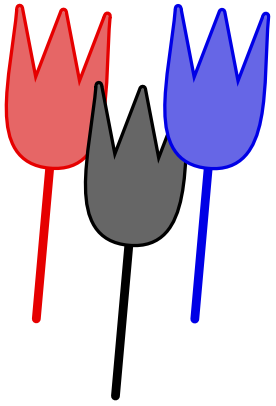
For easy alignment of the split pages, cut marks are added. This can be turned off by setting `marking` to `off`. The `n` parameters determine the number of cells per split off section, and the `d` parameters specify the overlap: a value of 1 means that each split off section has one row and/or columns in common with its neighbour.

The splitter can be used with the split float placement macro:



```
\splitfloat
  {\placefigure{What a big flowchart this is!}}
  {\FLOWcharts[mybigflow]}
```

Here every flowchart gets an caption with a decent subnumber.



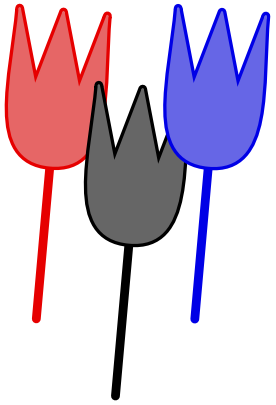
Other features

It is possible to predefine flow charts in a way similar to external figures. Currently this mechanism is under construction, so describing it would be a bit premature.

Crossing lines, which are often forbidden in charts, can be made less confusing by adding a gap in the lines to be crossed. This is one of the features that are already implemented but not yet accessible by the ConTeXt user interface.

Another feature, used in this document, concerns automatic down–scaling. As soon as we start scaling illustrations, we introduce inconsistent typography, especially in the bodyfontsize. Therefore, the flow chart macros, when told to, are able to automatically scale down in steps of 1 point.

This Up–To–Date was first published in GUST magazine and presented at the anual meeting of the Polish user group in Bachotek 1999. This (updated) version is a reprint of MAPS volume 23.



task force members	Tobias Burnus Gilbert van den Dobbelsteen Hans Hagen Taco Hoekwater
dedicated mailing list	ntg-context@ntg.nl
contacting authors	pragma@wxs.nl
examples, manuals and code	www.pragma-ade.nl www.ntg.nl/context frambach.eco.rug.nl/context
authors	Hans Hagen Ton Otten
processing date	September 29, 1999
current version	1999.9.17