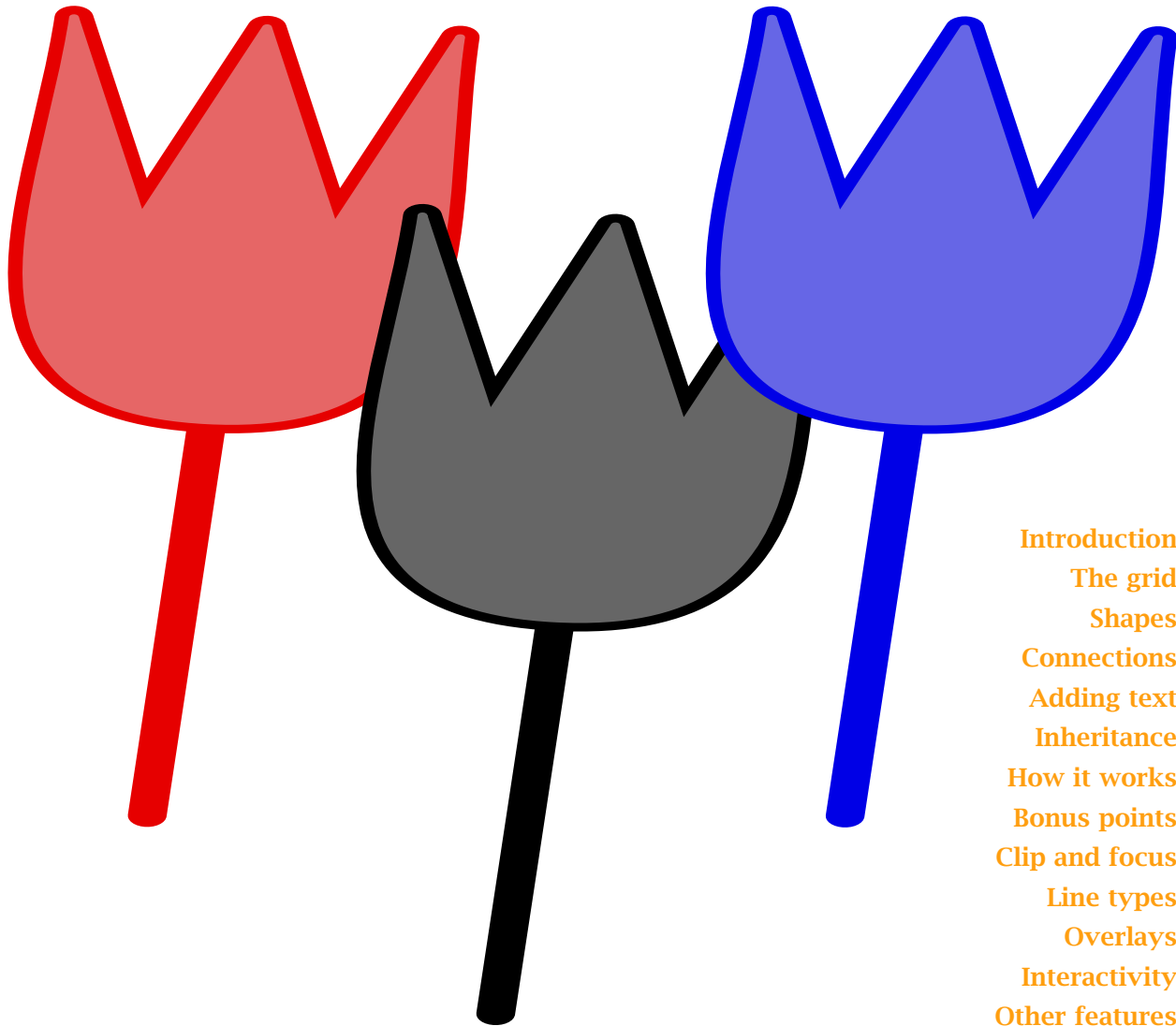


# CONTEXT

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**  
**Interactivity**  
**Other features**

PRAGMA ADE  
Ridderstraat 27  
8061GH Hasselt NL



# Introduction

This is just another story of  $\text{T}_{\text{E}}\text{X}$  meeting  $\text{METAPOST}$ . This time we will focus on charts, especially flow-charts. In  $\text{CONTEX}$ T flow-chart 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 is to 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 spend 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 unmistakably their advantages. As soon as  $\text{CONTEX}$ T 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 flow-charts. 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 support interactivity and were to be presented as a whole and in sub-chart 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. So there are our motives for using  $\text{T}_{\text{E}}\text{X}$  and  $\text{METAPOST}$  in favour of a dedicated drawing program.

# The grid

A flow-chart, or actually whatever chart we're dealing with, 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.

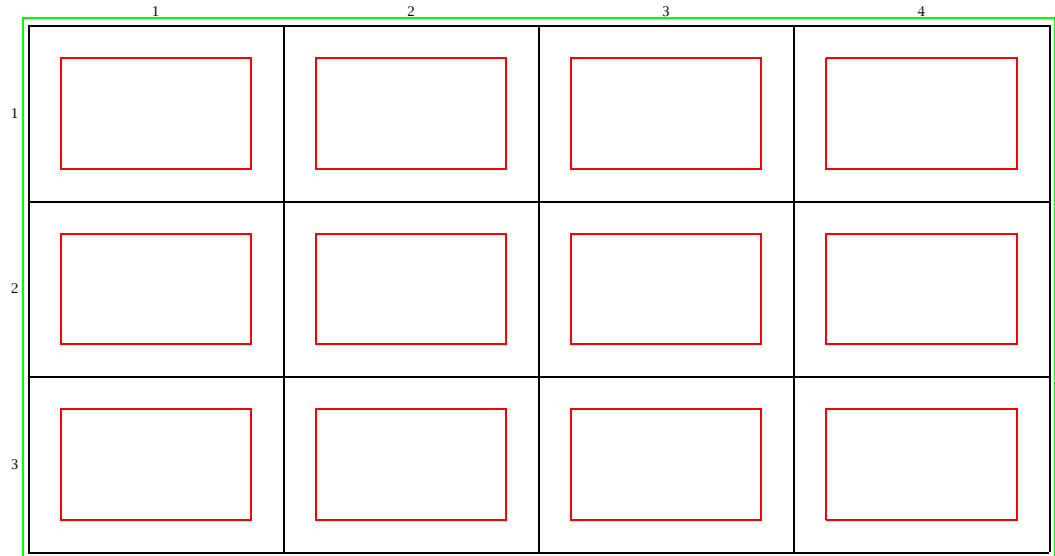
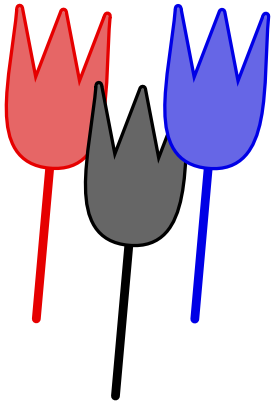
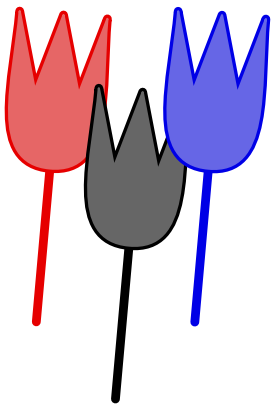


Figure 1 The grid.

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:

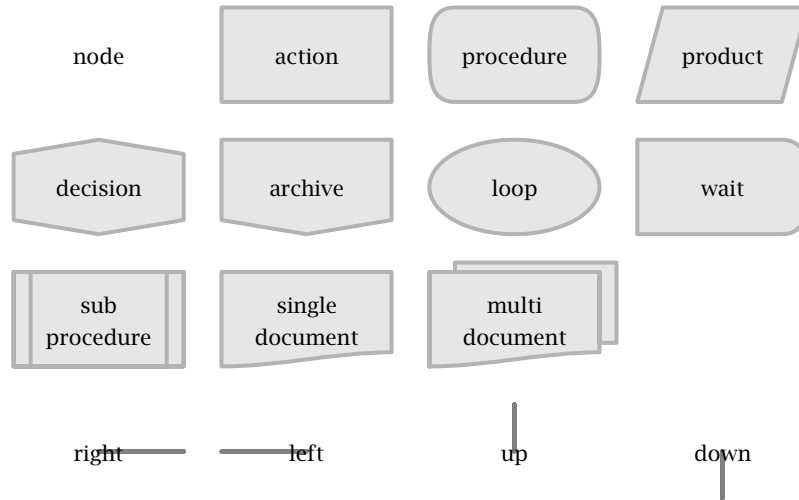
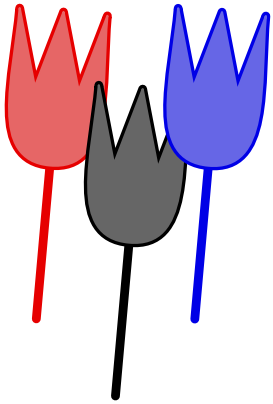
```
\FLOWchart[grid]
```

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 framed. Normally the something is a text. A shape has a frame, which can have a certain width and color, as well as a background. In this respect it acts like `\framed`. The shapes that make sense in flow-charts are identified by the names given in [figure 2](#). There are more shapes available, but they are identified by a number. The number of shapes will quite certainly increase.



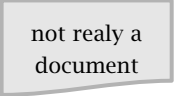
**Figure 2** The shapes.

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 is build out 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 T<sub>E</sub>X. Later we will see that there are some more fields to fill. Names are local to a chart.

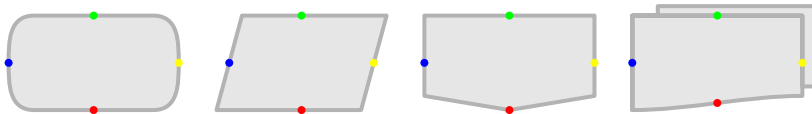


not realy a  
document

**Figure 3**

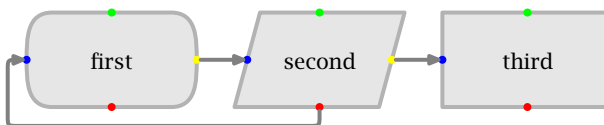
# Connections

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.



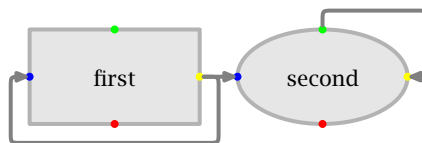
**Figure 4** The connection points.

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.

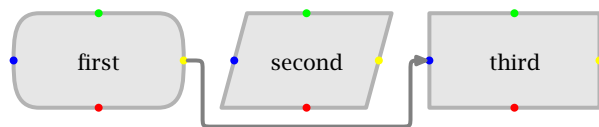


**Figure 5** A few connections.

There can be more than one connection per shape. When defining such a connection we first specify the direction. In this example `[r]` 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 where 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 a shorter route. Whenever possible, i.e. when no shapes are crossed, the routines will take a shortcut. I have to admit that the routines in itself are rather stupid, but for normal use they suffice.

In **figure 9** we see a bit ridiculous chart. Generally spoken, when two lines end at the same point, it makes sense to let them connect. When on the other hand lines originate at the same spot or cross halfway, readers can be confused. Therefore such lines are drawn in such a way that they not really touch.



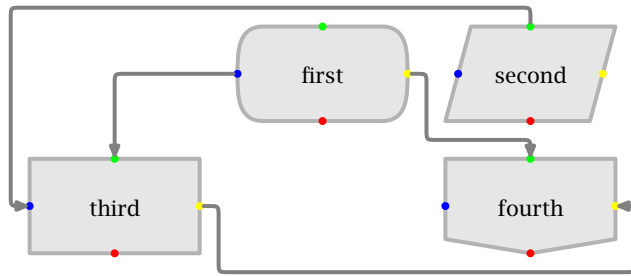
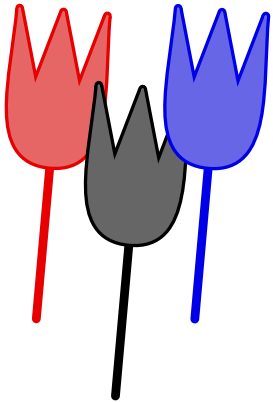


Figure 8 Following grid lines.

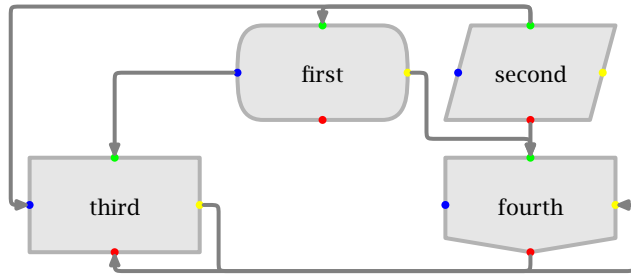
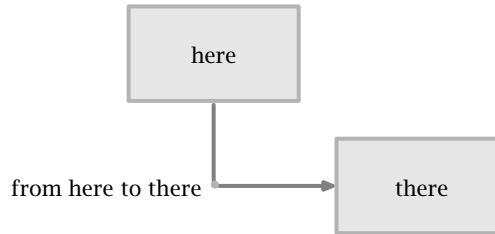


Figure 9 Confusing (crossing) grid lines.

# Adding text

In [figure 10](#) we have added some comment to a connection. Like the dots at the connections, the mid points show up in a special debugging mode. Comment will be placed relative to this point. In [figure 10](#) this is 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 8 locations, simply `l`, `r`, `t`, `b`, or a combination like `tr`.

```
\startFLOWce11
...
\comment [l] {from here to there}
...
\stopFLOWce11
```

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

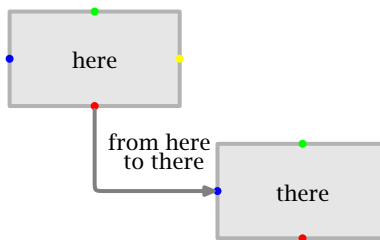


Figure 11 Labels to connection points.

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

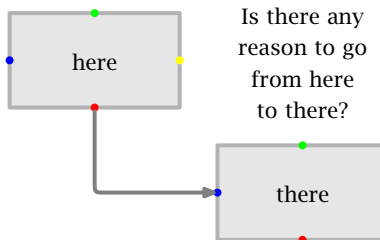


Figure 12 Text without shapes.

In [figure 12](#) we see some text without any shape 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, it happens that such a chart grows or comes in several sub-charts. To prevent us from retyping the same components again and again, we can split up the chart and combine these pieces. Inclusion is straightforward: the sub-chart 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.

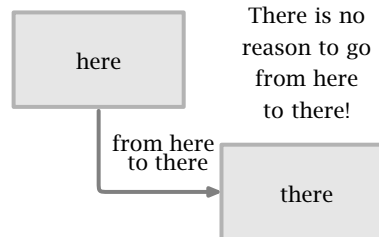


Figure 13 Sharing components.

# 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 between  $\TeX$  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 `CON $\TeX$ T`. Additional communication from `METAPOST` to `CON $\TeX$ T` is handled in the module itself. This rather fuzzy description is visualized in [figure 14](#). Depending on the circumstances, among the other processes involved are: color conversion and reduction and conversion to PDF. So, when watching this module in action, don't feel disturbed by the many steps involved.

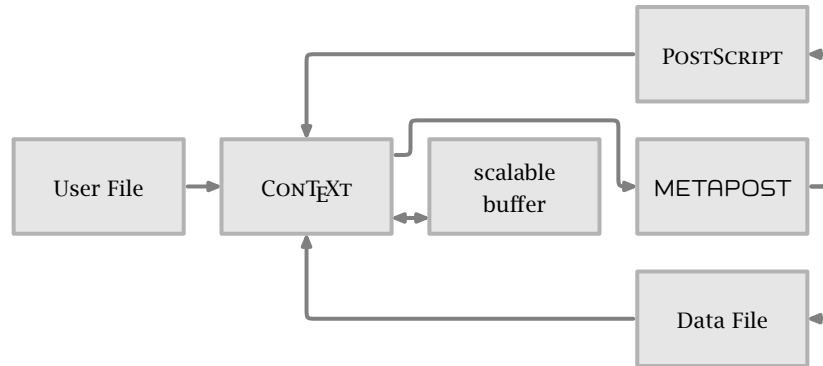
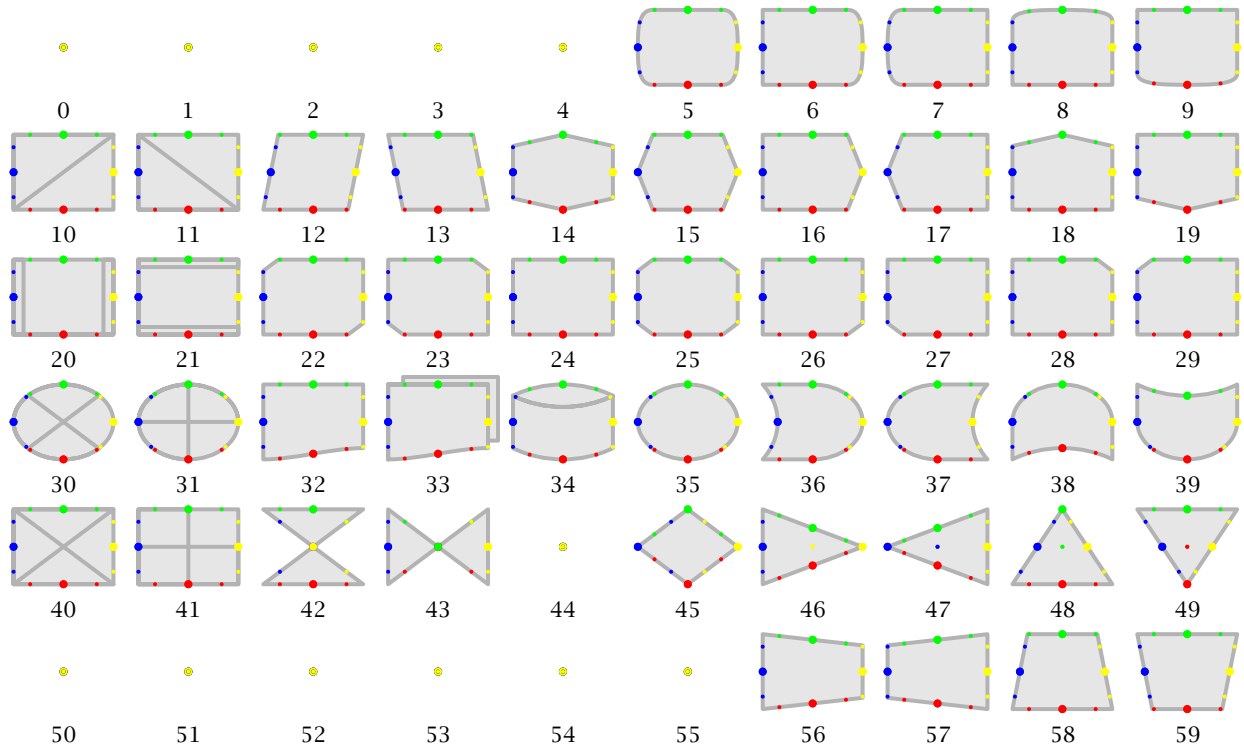
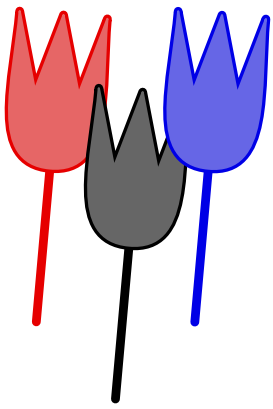


Figure 14 The process.

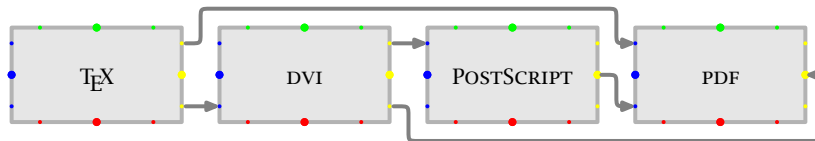
A few pages back we introduced the shapes by name. There are however some more shapes. Each shape can be reached by its number. In **figure 15** all available shapes are shown. The zero numbered shape is the official node shape, so don't use the other ones. Apart from shape 0 the other gaps are candidates for new shapes.



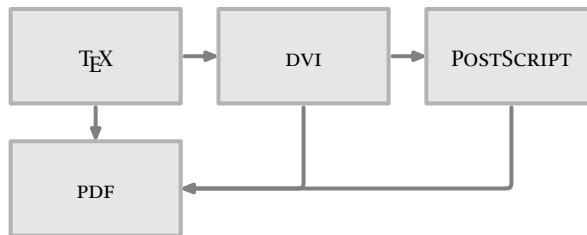
**Figure 15** All shapes by number.

# Bonus points

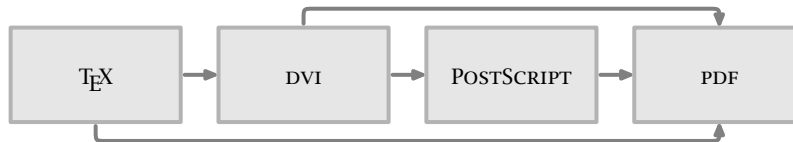
Sometimes charts can become pretty large. Also, charts can be complicated by the fact that more than one connection starts or ends at a shape. **Figure 16** shows a way out.



**Figure 16** Even more points.

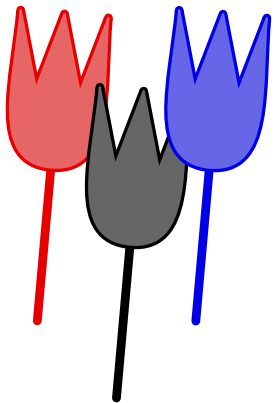


**Figure 17** An alternative for **figure 16**.



**Figure 18** Yet another 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 companions: positive and negative. In connections the left points are: p], l and n]: positive left, left and negative left, so the first cell in [figure 16](#) is defined as:

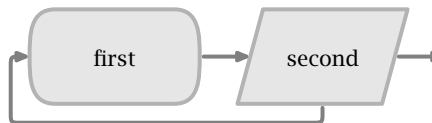
```
\startFLOWcell
  \name      {tex}
  \location  {1,1}
  \shape     {action}
  \text      {\TEX}
  \connection [prp] {pdf}
  \connection [nrn] {dvi}
\stopFLOWcell
```

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 that they are off-center. This does not free users from thinking about better ways to draw such a chart.

# Clip and focus

The flow-charter 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** is 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]
```

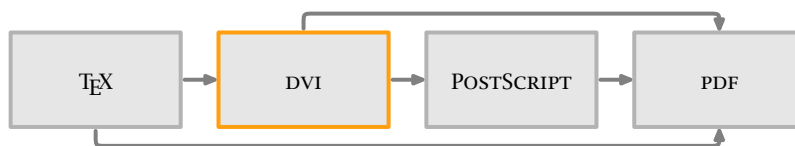


Figure 20 Gaining some focus.

```
\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:

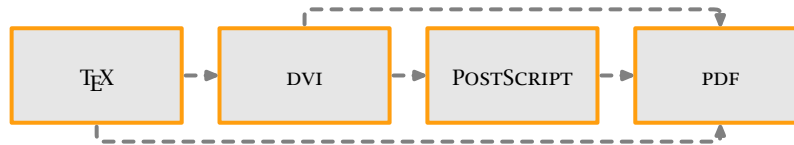
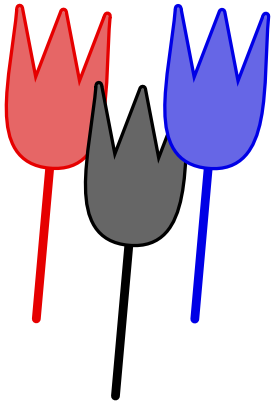


Figure 22 Dashed and colored lines.

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, one also has access to the graphic code.

```
\setupFLOWcharts[...]=...]
```

option	test
width	<i>dimension</i>
height	<i>dimension</i>
offset	<i>dimension</i>
dx	<i>dimension</i>
dy	<i>dimension</i>
nx	<i>number</i>
ny	<i>number</i>
x	<i>number</i>
y	<i>number</i>
autofocus	<i>name</i>
focus	<i>name</i>
backgroundcolor	<i>name</i> <u>white</u>



`\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 only? In `CONTEX`T most frames related features can have overlays. Because in this flow-chart 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.



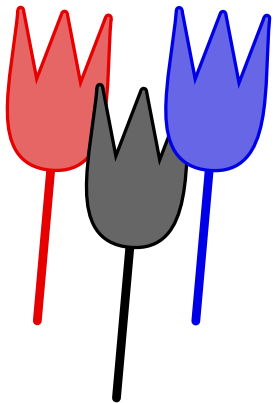
**Figure 23** Overlays.

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

```
\startFLOWce11
...
\figure {cow}
...
\stopFLOWce11
```

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

```
\startFLOWce11
...
\overlay {coward}
...
\stopFLOWce11
```



Here we call for an overlay, for instance defined by

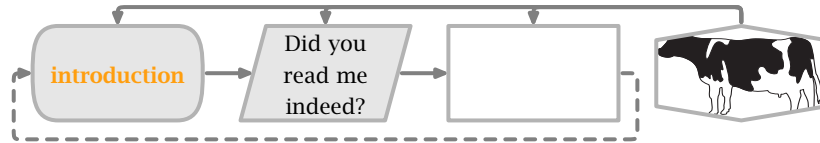
```
\defineoverlay  
[coward]  
[{\externalfigure  
[cow]  
[width=\overlaywidth,  
height=\overlayheight]}]
```

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

# Interactivity

One of the reasons for writing this module was that we wanted interactive flow-charts. The shape text can of course contain references.<sup>1</sup> The shape as a whole becomes a button as soon as we add the `\destination` entry:

```
\startFLOWce11
...
\destination {destination}
...
\stopFLOWce11
```



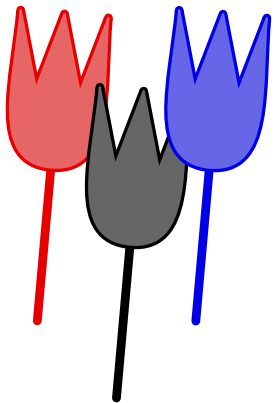
**Figure 24** Overlays, help and buttons.

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

```
\startFLOWce11
...
\text {\goto{introduction}[introduction]}
...
\stopFLOWce11
```

<sup>1</sup> 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
```

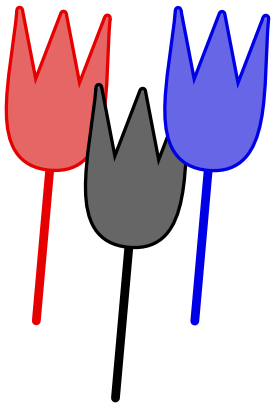
The third cell is a movie, which is only visible in suited viewers. The movie is included as an overlay.

```
\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:



```
\starthe\lptext [identifier]
```

```
... text ...
```

```
\stophe\lptext
```

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 flow-chart 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 change to interfere with other interactive things. Of course users should be aware of this feature.

So, we can put text in a shape, provide one or more overlays, turn cells and/or part of the text in a shape into hyper-links and show help when requested.

## Other features

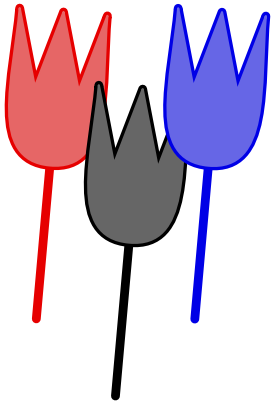
It is possible to predefine flow charts in a way similar to external figures. Currently this mechanism being optimized, so describing it would be a bit premature.

Crossing lines, which are often forbidden in charts, can be made less confusing by adding a gap in the lines to be crossed. Features like this are already implemented but not yet accessible by the `CONTEX` 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.

At this moment we are using and testing this module in typesetting a quality assurance manual of about 3000 pages and with over 1000 (sub)charts. As soon as the `TEX` macros and `METAPOST` code are stable and tested, this module will be added to the `CONTEX` distribution.

*This document will be updated when the module is finished.*



task force members Tobias Burnus  
Gilbert van den Dobbelsteen  
Hans Hagen  
Taco Hoekwater

dedicated mailing list [ntg-context@ntg.nl](mailto:ntg-context@ntg.nl)  
contacting authors [pragma@wxs.nl](mailto:pragma@wxs.nl)

examples, manuals and code [www.ntg.nl/context](http://www.ntg.nl/context)  
[frambach.eco.rug.nl/pragma](http://frambach.eco.rug.nl/pragma)  
[www.pragma-ade.nl](http://www.pragma-ade.nl)

authors Hans Hagen  
Ton Otten

processing date May 11, 1999  
current version 1999.5.10