Drawing trees in LaTeX

Wednesday, February 21st, 2007 | LaTeX

Last time we looked at how we could import graphics in the document using the graphics package. Today, we will look at how we can create vector graphics of trees from inside our LaTeX documents. As there are many aspects of the natural sciences that requires high quality drawings of trees, graphs, etc., it should come as no surprise that there are several packages for LaTeX trying to solve this issue. Some of these are:

- pstricks,
- mfpic,
- xypic, and
- PGF/TikZ

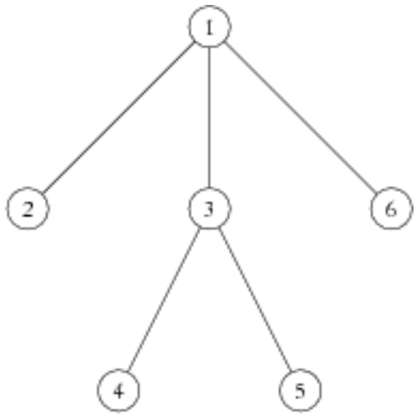
There are advantages and disadvantages to all the packages, naturally, mfpic depends on metapost, which takes a certain mindset to use, pstricks works only with latex and not with pdflatex, xypic is primarily for diagrams and graphs, but can also be used for other things, but my definite favourite is pgf, as it works with both latex and pdflatex and generates good-looking output, at least in my opinion.

The pgf package is split in two: an easy-to-use frontend called TikZ and the backend called pgf. Today we'll be looking exclusively at the frontend and try to draw simple trees with it.

First, let us look at an example of a tree:

```
\begin{tikzpicture}
    \tikzstyle{every node}=[circle,draw]
    \node {1}
        child { node {2} }
        child {
            node {3}
            child { node {4} }
            child { node {5} }
        }
        child { node {6} }
    ;
\end{tikzpicture}
```

What is happening here should be fairly self-evident, but let us run over it quickly. The call to \tikzstyle indicates that every node should be drawn, shaped as a circle. As for the tree, we have the root node with a 1 written in it. This root note has three children, of which the middle children has two further children. This gives us the following result (click on the image for the original PDF):



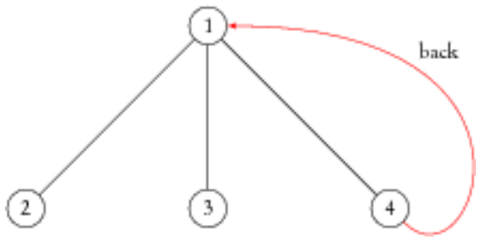
Another useful feature of TikZ is that you can name nodes so you can refer to them later on:

```
\begin{tikzpicture}
    \tikzstyle{every node}=[draw,circle]

    \node (root) {1}
        child { node {2} }
        child { node {3} }
        child { node (rightmost) {4} }
    ;

    \tikzstyle{every node}=[]

    \draw[-latex,color=red]
        (rightmost) .. controls +(southeast:1cm) and
                                +(right:3cm) ..
            node[near end,above right,color=black] {back}
        (root);

\end{tikzpicture}
```

In addition to figure 1, we can see that to name nodes we add a (name) after the node command. Once we've specified the actual tree, we reset the global style settings for nodes as we don't want our edge labels having a visible circle around them. The last draw command uses TikZ' syntax (which is alike to METAPOST's syntax) for drawing Bézier curves and the embedded node statement is for placing an edge label. The controls statement says that the first Bézier control point is placed 1cm to the south east of the (rightmost) node, and the second Bézier control point is placed 3cm to the right of the (root) node. Thus we can draw lines between nodes easily without explicitly having to state their absolute placement.

The southeast and right in the control nodes are called *anchor points* in TikZ, and there are plenty other of these. There are, in fact, so many means of customising nodes that it's impossible to cover it all here. Instead, use the PGF manual. It is very thorough and example-based, so you should be able to find solutions for pretty much everything.

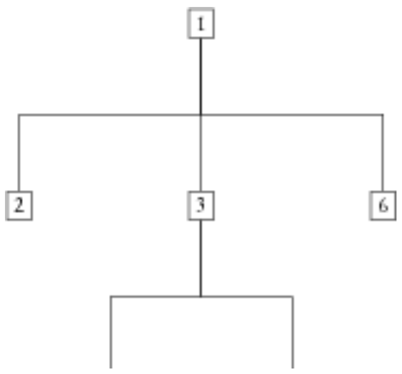
Using the above code snippet we get the following result:



Before we leave the topic of drawing trees with TikZ, let's look at the different forms of trees one can draw: trees growing downward (we've already seen these), trees growing upward, sideways, and trees where the edges fork down rather than go straight down, or where the edges curl down. There are plenty of possibilities. As the last example, let us look at the traditional fork down:

```
\begin{tikzpicture}
    \tikzstyle{every node}=[draw,rectangle]

    \node {1}
        [style=edge from parent fork down]
        child { node {2} }
        child {
            node {3}
            child { node {4} }
            child { node {5} }
        }
        child { node {6} }
    ;

\end{tikzpicture}
```

This requires that you add \usepackage{pgflibrarytikztrees} to your preamble, otherwise the *fork down* isn't available. Once you've done that, this is the result of compiling the drawing:



There is plenty more to the node aspect of TikZ. Indeed, it can also be used to create graphs quite elegantly, and, naturally, TikZ doesn't limit itself to just trees and graphs, it's a quite versatile vector drawing language inside LaTeX. In future posts we'll look at some more of these features.

Tags: LaTeX, programming

hstuart.dk

http://hstuart.dk/2007/02/21/drawing-trees-in-latex/

http://goo.gl/ffgS