# Creating Technical Posters With LaTeX

Nicola Talbot
http://theoval.cmp.uea.ac.uk/~nlct/

Tuesday 23rd January, 2007

## Contents

## 1 Introduction

In the field of research, conference papers sometimes have to be accompanied by a poster presentation. You have produced the article in LaTeX, but what about the poster presentation? It needs to be large, say A0, it needs to use a large font, the standard one or two column formats don't apply, and howabout a splash of colour? Can this be done in LaTeX? The answer is yes, and this document will show you how using a worked example.

This examples listed in the document can be downloaded from `http://theoval.cmp.uea.ac.uk/latex/posters/examples/`.

## 2 Setting the paper size and large fonts

The a0poster class file is well suited for creating large posters. It is based on the article class file, but uses large paper sizes and large fonts. The following class options govern the paper size and orientation:

| | |
|---|---|
| landscape | landscape (default) |
| portrait | portrait |
| a0b | “DIN A0 big” (default) |
| a0 | DIN A0 |
| a1 | DIN A1 |
| a2 | DIN A2 |
| a3 | DIN A3 |

The available font sizes and their declarations are shown in Table 1. You will need to make sure you use scalable fonts, as the standard Computer Modern fonts won’t be available in large sizes. For the Times/Helvetica/Courier combination, use the mathptmx, helvet and courier packages:

```
\usepackage{mathptmx}
\usepackage[scaled=.90]{helvet}
\usepackage{courier}
```

Table 1: Font sizes supplied by the a0poster class

| | |
|---|---|
| `\tiny` | 12pt |
| `\scriptsize` | 14.4pt |
| `\footnotesize` | 17.28pt |
| `\small` | 20.74pt |
| `\normalsize` | 24.88pt |
| `\large` | 29.86pt |
| `\Large` | 35.83pt |
| `\LARGE` | 43pt |
| `\huge` | 51.6pt |
| `\Huge` | 61.92pt |
| `\veryHuge` | 74.3pt |
| `\VeryHuge` | 89.16pt |
| `\VERYHuge` | 107pt |

## 3 Title and Section Headings: fonts and colours

A bit of colour makes a poster more interesting, BUT DON’T GO OVER THE TOP.

The easiest way to change the fonts and colours used in the title and section headings is to create a small package that redefines the relevant commands. For example, I’m going to make a poster with a `\Huge` bold sans-serif navy blue title, the names of the authors will appear in `\large` sans-serif cornflower blue and the (sub)section headings will all appear in sans-serif royal blue. To do this, I am going to use my text editor to create a file called `postercols.sty`, and type in the following:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{postercols}
\RequirePackage[dvipsnames]{xcolor}

\renewcommand\@maketitle{%
  \newpage
  \null
  \vskip 2em%
  \begin{center}%
  \let \footnote \thanks
    {% set colour and font for the title
    \color{NavyBlue}\sffamily\bfseries\Huge
    \@title \par}%
    \vskip 1.5em%
    {% set colour and font for the author
      \color{CornflowerBlue}\sffamily\large
      \lineskip .5em%
      \begin{tabular}[t]{c}%
        \@author
      \end{tabular}\par}%
    \vskip 1em%
    {% set colour and font for the date
    \color{CornflowerBlue}\sffamily\large \@date}%
  \end{center}%
  \par
  \vskip 1.5em}

\renewcommand\section{%
\@startsection{section}{1}{\z@}%
              {-3.5ex \@plus -1ex \@minus -.2ex}%
              {2.3ex \@plus.2ex}%
              {\color{RoyalBlue}\sffamily\Large\bfseries}}
\renewcommand\subsection{%
\@startsection{subsection}{2}{\z@}%
              {-3.25ex\@plus -1ex \@minus -.2ex}%
              {1.5ex \@plus .2ex}%
              {\color{RoyalBlue}\sffamily\large\bfseries}}

\endinput
```

**Notes**:

- All LaTeX $2_\varepsilon$ packages should start with `\NeedsTeXFormat{LaTeX2e}`
- You need to declare the name of the package. This is just the file name without the `.sty` extension: `\ProvidesPackage{postercols}`

- The xcolor package provides driver-independent access to several kinds of colour tints, shades, tones and colour mixtures. The option dvipsnames allows the use of the dvips colour names (e.g. RoyalBlue). So the fact that I want to use PDFLaTeX rather than LaTeX won't make a difference.

- I copied the definition of `\@maketitle` from the article class file (which is loaded by the a0poster class file) into `postercols.sty`, and edited it to use colour and fonts of my choice.

- Similarly, I copied the definition of `\section` and `\subsection` from the article class file into `postercols.sty` and edited the font declarations.

- My poster isn't going to go lower than sub-sections, so I haven't bothered to redefine `\subsubsection` and `\paragraph`, but they can be similiarly redefined.

- You should use the command `\endinput` to indicate the end of the package.

I then created a document called `poster.tex` that looked like:

```
\documentclass[a0,landscape]{a0poster}

\usepackage{mathptmx}
\usepackage[scaled=.90]{helvet}
\usepackage{courier}
\usepackage{postercols}

\title{A Sample Poster Created Using \LaTeX}
\author{Nicola Talbot\\
School of Computing Sciences\\
University of East Anglia\\
Norwich, Norfolk. NR4 7TJ
\and
Gavin Cawley\\
School of Computing Sciences\\
University of East Anglia\\
Norwich, Norfolk. NR4 7TJ}
\date{}

\begin{document}
\maketitle
\thispagestyle{empty}

\section{Introduction}
This is a sample poster.

\end{document}
```

## 4 Text Layout

As it stands, the `poster.tex` file created in the previous section is in a one column format, which isn't much use for a poster, and neither is the standard two-column format either. Most of my posters tend to have four columns, with maybe a figure or table spanning a couple of columns, and the title spanning all columns. This can be achieved using the flowfram package.

The flowfram package defines three types of frame: *flow* frames, *static* frames and *dynamic* frames. The flow frames are the principle type of frame. The document text will flow from one flow frame to the next in the order that the frames were defined, just as text flows from the left column to the right column in LaTeX's standard two-column mode. The contents of the static and dynamic frames need to be set explicitly using one of the available commands or environments. The contents of a static frame are stored in a savebox, and so are typeset only once, whereas the contents of dynamic frames are stored in a macro, and so are typeset everytime the frame is displayed. Since posters typically only have one page, this is something we don't need to worry about, however you do need to consider the "stacking order". This is the order in which all the frames are laid down on the page. First the static frames are placed on the page (starting with the first static frame to be defined), then the flow frames are placed on the page (starting with the first flow frame to be defined) and finally the dynamic frames are placed on the frame (starting with the first dynamic frame to be defined.) If two frames are positioned in the same location, the frame above will partially obscur the frame below.

For my example poster, I would like to lay out the contents as shown in Figure 1. The top frame is going to be a static frame, as it's only going to contain the title. The four long frames containing the document text will be flow frames. The frame containing the wide table will also be a static frame.

The flowfram package provides the following commands to create new frames:

```
\newstaticframe{<width>}{<height>}{<x>}{<y>}[<label>]
\newflowframe{<width>}{<height>}{<x>}{<y>}[<label>]
\newdynamicframe{<width>}{<height>}{<x>}{<y>}[<label>]
```

where <*width*> and <*height*> are the width and height of the frame and <*x*> and <*y*> are the $x$ and $y$ co-ordinates of the bottom left hand corner of the frame with respect to the bottom left hand corner of the typeblock.

It can be quite fiddly trying to determine the dimensions and co-ordinates, but fortunately the flowfram package provides some commands that create predefined layouts. One of these commands is:

```
\Ncolumntop{<type>}{<n>}{<H>}
```

This command creates an <*n*> column layout with a <*type*> frame of height <*H*> across the top, where <*type*> is one of: `static`, `flow` or `dynamic`. This is very similar to the layout for our example. The vertical gap between the header frame and the column frames is given by `\vcolumnsep` and the gap between the columns is given by

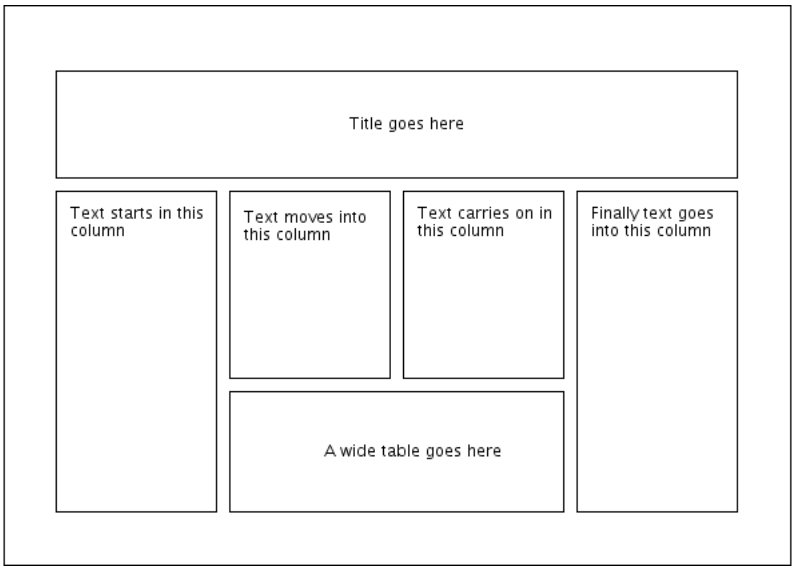
Title goes here
Text starts in this column
Text moves into this column
Text carries on in this column
Finally text goes into this column
A wide table goes here

Figure 1: Example Poster Layout

`\columnsep`. These lengths are both 10pt by default, which is too small for such a large poster. Therefore I am going to change both these values to `\baselineskip`:

```
\setlength{\vcolumnsep}{\baselineskip}
\setlength{\columnsep}{\vcolumnsep}
```

We can now create a four column layout with a 4in high static frame for the title:

```
\Ncolumntop{static}{4}{4in}
```

next we can modify the two middle frames to make them shorter. Each frame has an identification number (IDN) unique to that frame type, and in this case the two middle frames have IDNs 2 and 3 (since they were the second and third flow frames to be created.) The height and $y$ co-ordinate can be changed using

`\setflowframe{`<*idn-list*>`}{`<*key=val list*>`}`

I want the frame containing the wide table to be 5in high, so the $y$ co-ordinate for the middle flow frames needs to be 5in plus a small gap, and the height of the middle flow frames will need to be shortened by that amount. The poster would look more balanced if this gap is the same as the gap between the header static frame and the flow frames below it, i.e. `\vcolumnsep`. So first define a new length, and set it to 5in + `\vcolumnsep`:

```
\newlength\offset
\setlength{\offset}{5in}
\addtolength{\offset}{\vcolumsep}
```

This gives us the new $y$ co-ordinate for the two middle frames, but how do we work out the height? You can find out the bounds of a flow frame, or a selection of flow frames, using:

```
\computeflowframearea{<IDN list>}
```

This command stores the bounds in the lengths `\ffareawidth`, `\ffareaheight`, `\ffareax` and `\ffareay`. So to work out the new height for the middle frames:

```
\computeflowframearea{2,3}
\addtolength{\ffareaheight}{-\offset}
```

Now reset the bounds of the two middle frames using these new values:

```
\setflowframe{2,3}{y=\offset,height=\ffareaheight}
```

Finally we need to add a static frame that spans the two middle frames. Earlier we used

```
\computeflowframearea{2,3}
```

This computed the entire area covered by the two middle frames, so the length `\ffareawidth` now stores the total width covered by the two frames. This will also be the width of the new static frame. We've already decided that the height of this new frame should be 5in, and the $y$ co-ordinate will be 0in, so that its base lies level with the two end flow frames. The $x$ co-ordinate should be the same as the $x$ co-ordinate of the second flow frame, which is currently stored in `\ffareax`. So we now have all the information to make the new static frame:

```
\newstaticframe{\ffareawidth}{5in}{\ffareax}{0in}[table]
```

I gave the static frame the label `table` so that I don't have to remember its IDN. I can also assign a label to the first static frame (that was created using `\Ncolumntop`)

```
\setstaticframe{1}{label={title}}
```

You can use the staticcontents or staticcontents* environment to set the contents of a static frame. If the frame has a label you can use the starred version, otherwise you must use the non-starred form with the IDN. So, to put the title into the static frame labelled `title`:

```
\begin{staticcontents*}{title}
\maketitle
\end{staticcontents*}
```

You can't put floats into a static or dynamic frame, but the flowfram package provides the alternative environments staticfigure and statictable. These environments may have a caption and label, but as their names suggest, they don't float. So to put a table into the static frame labelled `table`:

```
\begin{staticcontents*}{table}
\begin{statictable}
\caption{A very wide table that spans two columns}
\label{tab:wide}
% table contents
\end{statictable}
\end{staticcontents*}
```

At the moment the frames don't have any borders, you can add a plain border using the `border` key in `\setflowframe` and `\setstaticframe`. Since I want to add a border to all the frames, I shall use `\setallflowframes` and `\setallstaticframes`:

```
\setallflowframes{border=plain}
\setallstaticframes{border=plain}
```

Putting it all together:

```
\documentclass[a0,landscape]{a0poster}

\usepackage{mathptmx}
\usepackage[scaled=.90]{helvet}
\usepackage{courier}
\usepackage{postercols}
\usepackage{flowfram}

\setlength{\vcolumnsep}{\baselineskip}
\setlength{\columnsep}{\vcolumnsep}

\Ncolumntop{static}{4}{4in}
\setstaticframe{1}{label={title}}

\newlength\offset
\setlength{\offset}{5in}
\addtolength{\offset}{\vcolumnsep}
\computeflowframearea{2,3}
\addtolength{\ffareaheight}{-\offset}

\setflowframe{2,3}{y=\offset,height=\ffareaheight}

\newstaticframe{\ffareawidth}{5in}{\ffareax}{0in}[table]
\setstaticframe{2}{clear}

\setallflowframes{border=plain}
\setallstaticframes{border=plain}

\title{A Sample Poster Created Using \LaTeX}
```

```
\author{Nicola Talbot\\
School of Computing Sciences\\
University of East Anglia\\
Norwich, Norfolk. NR4 7TJ
\and
Gavin Cawley\\
School of Computing Sciences\\
University of East Anglia\\
Norwich, Norfolk. NR4 7TJ}
\date{}

\begin{document}
\begin{staticcontents*}{title}
\maketitle
\end{staticcontents*}
\thispagestyle{empty}

\section{Introduction}

This is the main body of the poster.

\subsection{A Sub Section}
Here's a sub section that references the wide
table, see Table~\ref{tab:wide}.

% this just fills the empty columns
\framebreak\mbox{}\framebreak\mbox{}\framebreak\mbox{}

\begin{staticcontents*}{table}
\begin{statictable}
\caption{A very wide table that spans two columns}
\label{tab:wide}
% table contents
\end{statictable}
\end{staticcontents*}

\end{document}
```

The flowfram package sometimes produces extra pages if you don't fill all the flow frames defined on the page. You can either ignore it while you are still editing your document, or, if you find that you have finished your document with one or more frames to spare, you can insert

```
\framebreak\mbox{}
```

for each spare frame.

## 5 Using a GUI to construct frames

If you find the idea of trying to determine the area and position of frames a bit daunting, or if you'd like to create fancy borders around your frames, but find it too tricky to do in LaTeX, you may prefer to use a graphical user interface (GUI) to construct the frames, and possibly draw in the borders, and add colour to the backgrounds. The JpgfDraw[1] application can be used for this purpose. Although principally designed as a graphics application, you can also use it to create a LaTeX $2_\varepsilon$ package based on the flowfram package. JpgfDraw's user manual has a step-by-step example[2] that creates a poster.

If you are put off using a GUI because you find it difficult to use a mouse, the latest version of JpgfDraw provides keyboard alternatives. You will need to type in all the co-ordinates, but you can instantly see the effects.

[1] http://theoval.cmp.uea.ac.uk/ nlct/jpgfdraw/

[2] http://theoval.cmp.uea.ac.uk/ nlct/jpgfdraw/manual/postertutorial.html

# Index