Batch Commander: A graphical user interface for TEX

Kaveh Bazargan
River Valley Technologies
kaveh@river-valley.com

Introduction

A constant criticism of TEX is that it is not user-friendly. In today's computer environment, users expect to be able to click buttons and choose menus, and to see a result immediately. Of course we know that TEX is a mark-up language, and there is no way that all of TEX's power can be accessed via menus and buttons. But with some limitations, it turns out that a graphical user interface (GUI) can be useful. In this article I will describe my attempt at such a GUI, which I have called Batch Commander.

Why Batch Commander?

When I first started on this project, I chose a name with 'TEX' in it. Then I realised that the GUI need not be restricted to TEX, but can be used with other programs that take text as input and produce graphical output. (The TEX preview is indeed a graphical output, even though it contains mainly text.) I have already used the GUI successfully with Povray (`http://www.povray.org`), and I believe it can be used with MetaPost.

Using the Batch Commander with different programs is simple, given the appropriate setup. When the main file is chosen from the pop-up menu, the file extension tells Batch Commander which program should be used. If the extension is `.tex`, then it knows that the config file (we'll come to this later) must have a `.sty` extension, and that the file should run through TEX. But if it has a `.pov` extension, then it will use `.inc` for the extension of the config file, and run Povray on the main file.

Limitations

Let's look at the limitations within which we will be working.

Use only global controls Batch Commander will only be used to apply *global*, as opposed to *local* controls. Thus, we will be able to change parameters such as `\baselineskip` and `\textwidth`, which apply to the whole document, but we cannot use a command like `\emph{}`, which applies to part of the document. So in principle, we can design a whole class file, but we cannot apply a style to a particular section of the text.

Use LATEX Although the program I will describe can be applied to any TEX program, e.g. plain TEX or ConTEXt, I will only consider a standard LATEX file. We will use the predictable structure of a LATEX file to our advantage. We will be able to load packages and modify the options, and we'll also be able to load any TEX command or primitive just after the `\begin{document}`.

File structure

Let us consider the minimal LATEX file shown in figure 1(a). There are two areas of 'global controls' which affect the output in this file:

1. The packages loaded, and their options;
2. (LA)TEX commands such as `\baselineskip...`, which come after the `\begin{document}`.

Let us make this file simpler by putting all global controls into another style file. Figure 1(b) shows this simplification by introducing another file called `river_valley.sty`. (The `\AtBeginDocument{...}` command ensures that anything enclosed within the braces is read only after `\begin{document}`.)

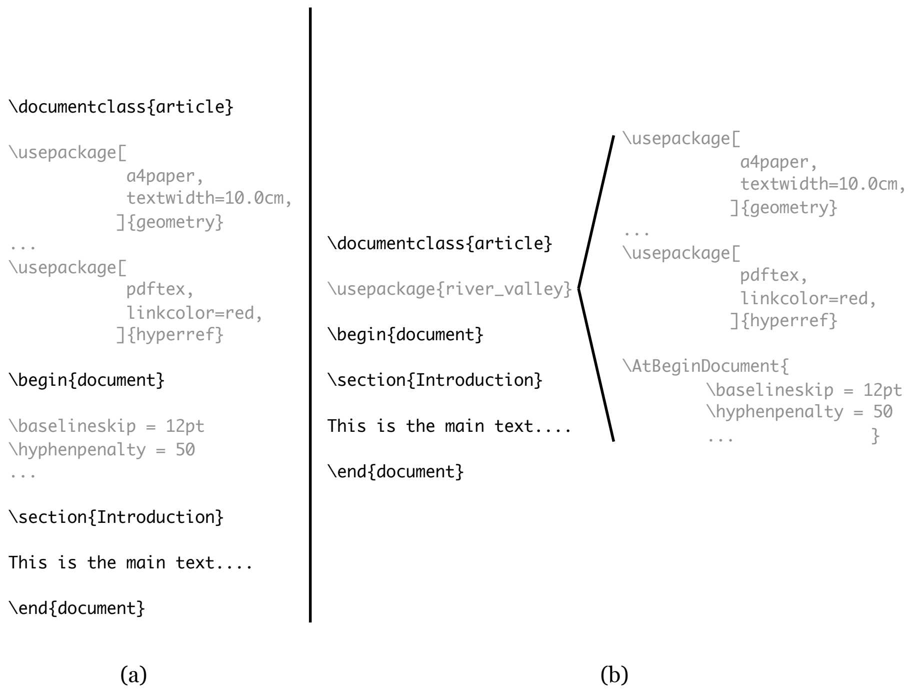
Separating the document file from the 'controlling' file means that once we have decided on a set of controls, they can be easily applied to other documents.

The overall concept

Our goal is to have a GUI that allows us to control of the contents of `river_valley.sty` interactively, have the file saved to disk, run the main file through TEX immediately, and finally show the preview.

The operating system I have used is Macintosh OS X (Tiger). As far as possible I have tried to keep the components of the system platform independent, so that it can be ported easily to other systems.

Controls Figure 2 shows the process of typesetting a file using the controls of a GUI. By 'controls' we mean graphical objects such as buttons and menus which take the place of editing a text file. This is how the process works:



```
\documentclass{article}

\usepackage[
           a4paper,
           textwidth=10.0cm,
          ]{geometry}
...
\usepackage[
           pdftex,
           linkcolor=red,
          ]{hyperref}

\begin{document}

\baselineskip = 12pt
\hyphenpenalty = 50
...

\section{Introduction}

This is the main text....

\end{document}
```

(a)

```
\documentclass{article}

\usepackage{river_valley}

\begin{document}

\section{Introduction}

This is the main text....

\end{document}
```

```
\usepackage[
           a4paper,
           textwidth=10.0cm,
          ]{geometry}
...
\usepackage[
           pdftex,
           linkcolor=red,
          ]{hyperref}

\AtBeginDocument{
        \baselineskip = 12pt
        \hyphenpenalty = 50
        ...            }
```

(b)

Figure 1: Simplifying the main file by putting all 'controls' in an external style file. (a) The original file; (b) the new main file which reads in an external 'controlling' file at run time.

- The user makes a change to a control, e.g. types in a number, or clicks a checkbox;
- the system immediately writes out a 'config' file, in our case '`river-valley.sty`';
- the main file is run through TEX,
- the screen preview is shown.

The idea is that the user sees only the controls, and the final screen preview, not the intermediate text files, unless he/she expressly wishes to.

Programming tools

The GUI The GUI needs to be completely flexible, and have the following facilities:

- Buttons, pop-up menus, and a rich mix of other interactive features to allow efficient control of values for parameters, choice of options, etc.
- ability to read and write text files;
- ability to communicate with other programs;
- be easy to program
- be easily portable to different platforms.

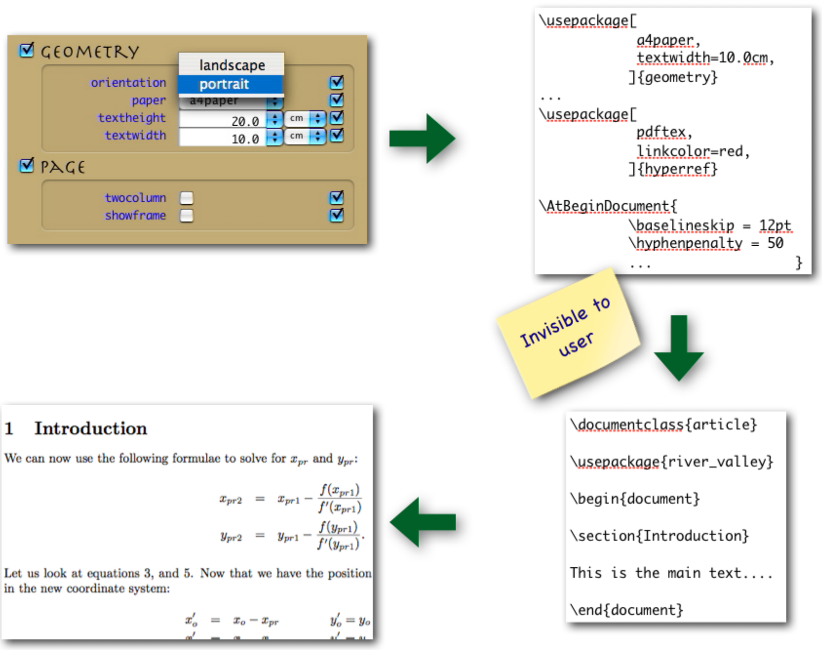
I chose the Runtime Revolution product (`http://www.runrev.com`) for the development environment. Revolution is a successor of Apple's HyperCard, which was a highly innovative scripting program written by Bill Atkinson, but which was neglected by Apple over the years, and effectively died a slow death. (In my opinion the 'killing off' of HyperCard was one of Apple's worst decisions.) My familiarity with HyperCard allowed me a quick start in Revolution, and it has worked extremely well so far, with no major drawbacks.

An advantage of Revolution is that it is cross-platform, so the majority of the work writing the GUI need not be duplicated for other platforms.

Communication of Batch Commander with TEX After the interactive changes have been made, TEX needs to be told to run the file and to show the preview. For this stage I used the scripting language of Revolution, i.e. Transcript, together with a mixture of shell scripts and AppleScript. This is the main area where each platform would need its own support.

TEX implementation I used pdfTEX exclusively for typesetting the TEX files. So when a document is typeset, a PDF file is produced in a single step. One reason for this is that we want to embed PDF-specific items, and we want to test that they are recorded correctly. pdfTEX allows the fastest route to the generation of PDF.

TEX preview I found TeXShop the most convenient way of viewing the PDF file, with minimal time



```
\usepackage[
            a4paper,
            textwidth=10.0cm,
           ]{geometry}
...
\usepackage[
            pdftex,
            linkcolor=red,
           ]{hyperref}

\AtBeginDocument{
           \baselineskip = 12pt
           \hyphenpenalty = 50
           ...                 }
```

```
\documentclass{article}

\usepackage{river_valley}

\begin{document}

\section{Introduction}

This is the main text....

\end{document}
```

1 Introduction

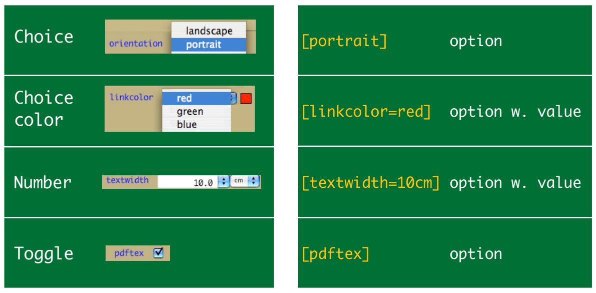
We can now use the following formulae to solve for x_{pr} and y_{pr}:

$$x_{pr2} = x_{pr1} - \frac{f(x_{pr1})}{f'(x_{pr1})}$$
$$y_{pr2} = y_{pr1} - \frac{f(y_{pr1})}{f'(y_{pr1})}.$$

Let us look at equations 3, and 5. Now that we have the position in the new coordinate system:

$$x'_o = x_o - x_{pr} \qquad y'_o = y_o$$

Figure 2: The process of using a GUI to modify the global parameters of a typeset document.


orientation
landscape
portrait
linkcolor
red
green
blue
textwidth
10.0
cm
pdftex

Choice	[portrait]	option
Choice color	[linkcolor=red]	option w. value
Number	[textwidth=10cm]	option w. value
Toggle	[pdftex]	option

Figure 3: Classification of 'controls' according to selection method, and output form.

delay and screen flicker. But in order to check hyperlinks, I used Adobe Acrobat Standard 7.0. A checkbox in Batch Commander allows this choice.

A detailed look at controls

When we examine what controls should look like in Batch Commander, and what they should look like in the output, i.e. in the config file, we find that they can be classified in two ways, according to (1) how they should look in the GUI, and (2) how they should appear in the config file. Figure 3 shows some examples of controls. Let us examine the four controls shown:

1. The first line shows a control that uses a pop-up menu to allow a choice between '`landscape`' and '`portrait`'. We call this type of control '`choice`'. In the config file, it is normally given as one of the options to a style, i.e. in square brackets. We will call such an output form '`option`'.
2. The second line allows selection from a list of colors, and might have the color shown next to it. This is similar to `choice`, and we call it '`choice color`'. The output is not just the selected item, as in the first line, but has to be given as a key-value pair. We call this output form '`option with value`'.
3. The third line is very different, in that a value needs to be entered into a field. There may also be a unit to append to the value, as in this case. So the user chooses a value, and selects the unit too. We can call this type of control a '`Number`'. The output is as the previous case, i.e. '`option with value`'. As there is a unit assigned in this case, this is appended.
4. The fourth line shows yet another type of control, namely an on/off button. This we call a '`toggle`'. In this case the output form is again `option`.

The point to note is that each control can have different output forms, but the same type on the GUI. Equally, two controls might have different forms for selection, but the same output form. We

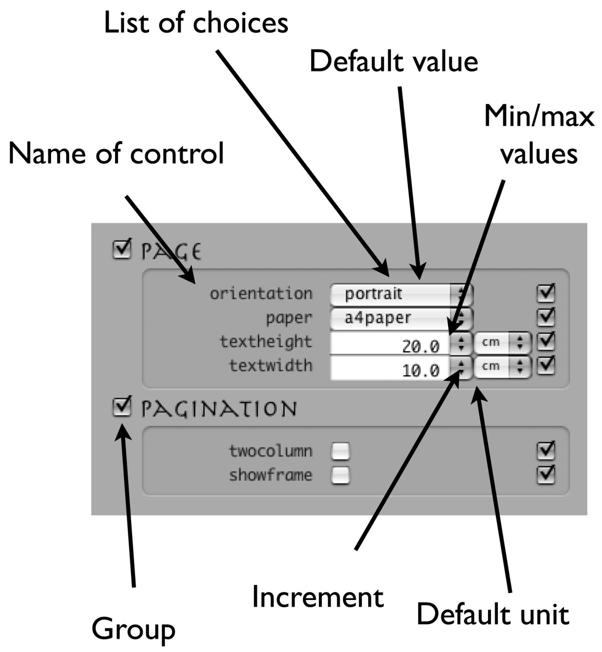


Figure 4: Attributes of controls.

will see later that this classification is useful in generating the controls automatically.

Having looked at the type of the control and the output form, let us look at other attributes of a control, so that we can define clearly what a control should look like. Figure 4 shows some attributes which a control might have. Here is a comprehensive list of the possible attributes of a control:

Group When there are a lot of controls, it is convenient to group them together logically, so that only a selected number of controls are visible at any time. So we should make each control be associated with a particular group.

Name Each control has a name. Depending on the output form, the name may or may not be written to the config file.

Description It is useful to have a short description of each control, in order to remind the user what the control does. This might be shown permanently on the GUI, or might be a tooltip or a pop-up field.

Selection type This is what we discussed above. It might, for example, be `choice`, `number`, or `toggle`.

Output form As discussed above, the output form determines how the data corresponding to the control is written to the config file, e.g. `option`.

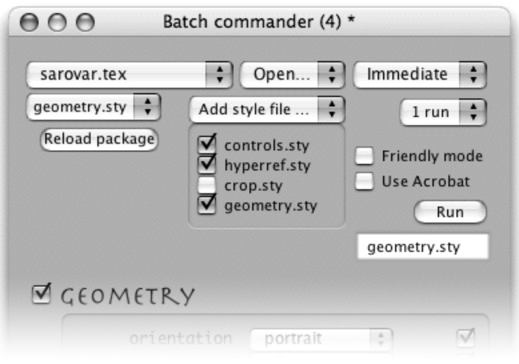


Figure 5: Main anatomy of Batch Commander.

Position The data from each control is normally written either as an option to a style file, within square brackets, or in an `\AtBeginDocument{...}` command. So the value given to position is either `package` or `begindoc`.

Unit A control may or may not have a unit associated with it.

Minimum and maximum values These apply in the case of `number` controls, where a value needs to be chosen from a range of numbers.

Choices If the selection type is `choice`, then the list of possible choices must be specified.

Default value This is the default, either of a numerical value, or from a list of choices. If the user does not interact with the controls, this is the value written.

Increment For numerical values, this is the increment between successive values offered to the user as choices.

Decimals This determines how many decimal units are displayed for numerical values of a particular control.

General anatomy of the GUI

Figure 5 shows the main overall control area of Batch Commander. The controls for a specific style file appear below this area. By looking at these overall controls, we can get a feel for the functionality of the GUI.

The top left pop-up button selects the main LaTeX file which is to be typeset (figure 6). This file will be opened, and all current settings will be applied to it. If a file is not in the list of available

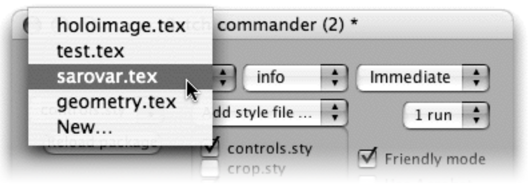


Figure 6: Choosing the main LATEX file.

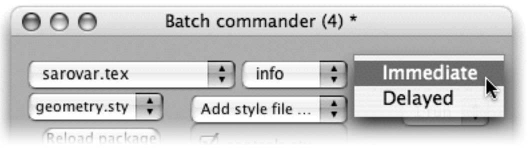


Figure 7: Immediate or delayed application of changes.

.tex files, then by choosing New..., the file and its path will be added to the list.

The top right button determines whether any changes made should be applied immediately to the config file (river_valley.sty), or only after the user clicks the run button (figure 7). (On slower systems, the Delayed option is best.) With Immediate selected, as soon as any control is clicked, a new config file is written, and the LATEX file is run to show the preview. The style management button is used to make style files available to the user (figure 8).

When a style file is available, it is included in the list of checkboxes just below it. Whether it is 'loaded', i.e. included in the config file, depends on whether the checkbox is ticked (figure 9). Furthermore, the style file will be written out in the order that they appear in the GUI, i.e. controls.sty first, and geometry.sty last. There is an intuitive mechanism to reorder the styles, by simply dragging them on the screen.

If the Friendly mode button is not checked, then each control for a style will have the actual name of the control shown, i.e. the name that will be written to the config file, and holding the mouse over this name shows the 'description', i.e. a short explanation of what the control does. Checking the Friendly mode button reverses this mode. See figures 10 and 11. This is simply a user preference which does not affect the functionality of the GUI.

The Use Acrobat button, when checked, shows the typeset preview using Adobe Acrobat Reader. Otherwise TeXShop is used for preview.

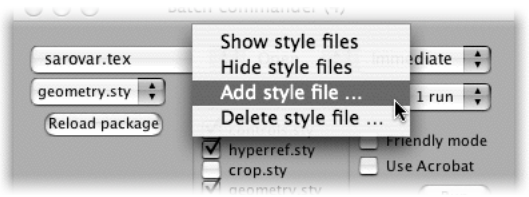


Figure 8: Managing style files available to the user.

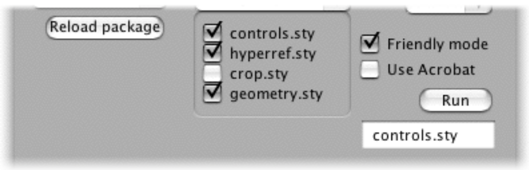
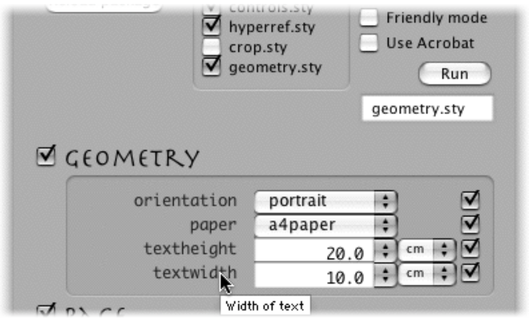
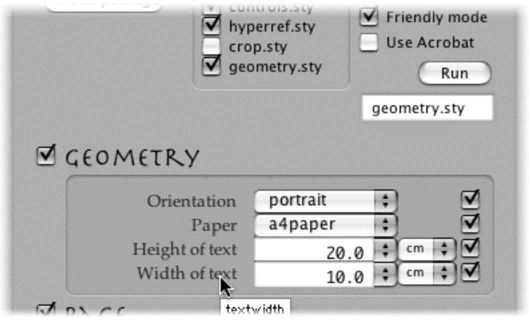


Figure 9: Four style files are available to the user. The three that are checked will be written to the config file, together with their options.



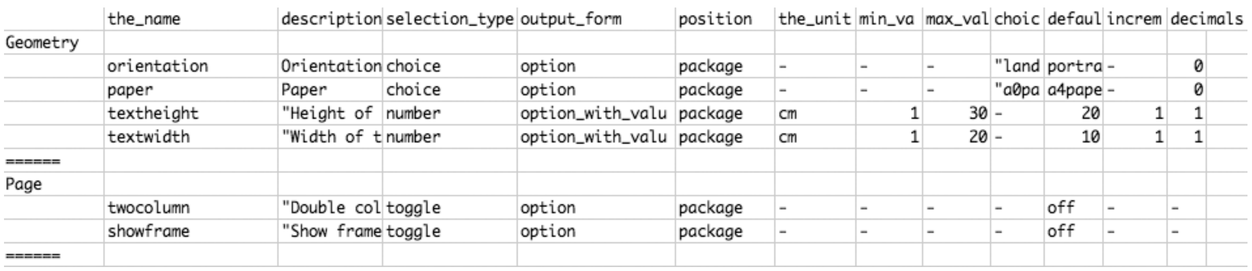
orientation portrait
paper a4paper
textheight 20.0 cm
textwidth 10.0 cm

Figure 10: Normal mode.



Orientation portrait
Paper a4paper
Height of text 20.0 cm
Width of text 10.0 cm

Figure 11: Friendly mode.



	the_name	description	selection_type	output_form	position	the_unit	min_va	max_val	choic	defaul	increm	decimals
Geometry												
	orientation	Orientation	choice	option	package	-	-	-	"land	portra	-	0
	paper	Paper	choice	option	package	-	-	-	"a0pa	a4pape	-	0
	textheight	"Height of	number	option_with_valu	package	cm	1	30	-	20	1	1
	textwidth	"Width of t	number	option_with_valu	package	cm	1	20	-	10	1	1
======												
Page												
	twocolumn	"Double col	toggle	option	package	-	-	-	-	off	-	-
	showframe	"Show frame	toggle	option	package	-	-	-	-	off	-	-
======												

Figure 12: A data file for `geometry.sty`. The set of controls to be included can be easily modified.

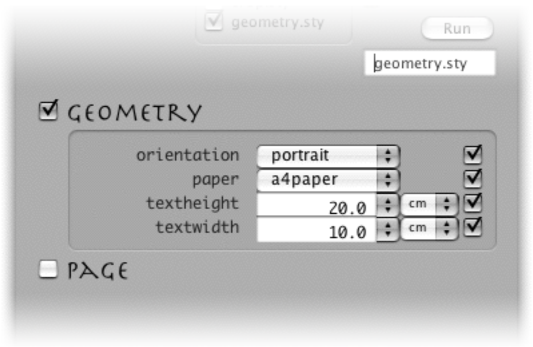


Figure 13: The controls generated using `geometry.data` shown in figure 12. Notice that only the ‘geometry’ group is checked and visible.

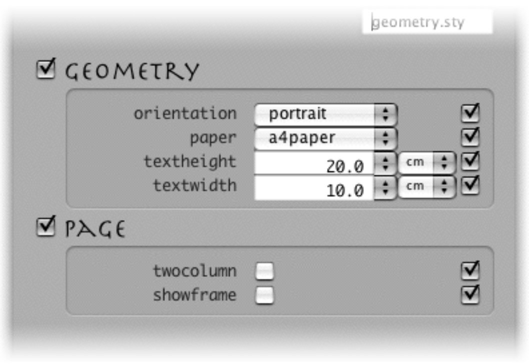


Figure 14: As figure 13, but with both ‘geometry’ and ‘Page’ groups shown

Generating the style controls

The underlying idea is that each style file has an associated set of controls. So we want it to be easy to create a set of controls when a style file is ‘loaded’.

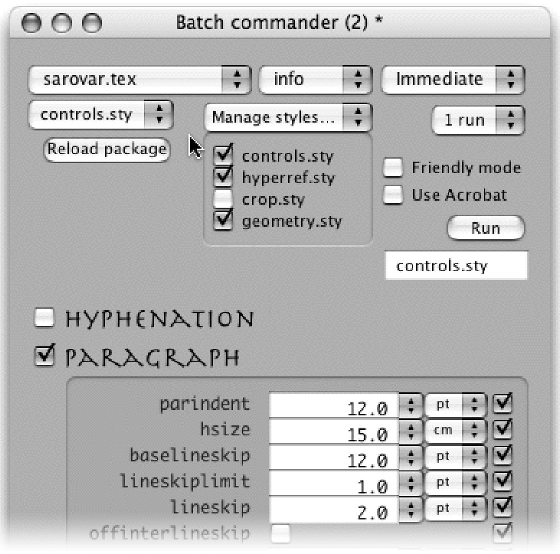
The method I have used is to create a `.data` file which has all the information about the control. This file is read by Batch Commander when a style file is loaded, and the controls are created immediately. The data file is simply a tab-delimited text file which contains a list of the controls, with all the attributes discussed above.

Figure 12 shows an example data file; in this case, the file is `geometry.data`, containing the control data for `geometry.sty`. It is important to note that the `.data` file is not definitive for each style file, and is ‘designed’ by the person who writes it. In this case I decided that the six entries under the `the_name` column were the controls I needed for my purposes. Another user might require a different set of controls.

The data file The way the data file is read in and interpreted can be best understood by examining figure 12, and the resulting GUI pages shown in figures 13 and 14. (To retain a reasonable text size, some of the items have been truncated, but it should be obvious what they are.) Here are some features of the data file:

- The first column is reserved for the group name. All rows between a group name and the first occurrence of ‘`======`’ are considered to be in that group. By grouping controls together, we can show only those groups we are interested in at one time.
- Apart from the first column, which denotes the grouping, all other columns can be written in any order. The system reads the title of the column in the first row, and thereafter assigns the correct attribute to each control.
- A single dash in a cell denotes ‘not applicable’. For example a toggle control cannot have minimum and maximum values. There must be no empty columns.
- If a `description` is more than one word, then it must be enclosed in double quotation marks.

The data file needs to be written manually, but only once for each style file.



[✓] controls.sty
[✓] hyperref.sty
[ ] crop.sty
[✓] geometry.sty

[ ] Friendly mode
[ ] Use Acrobat

controls.sty

[ ] HYPHENATION
[✓] PARAGRAPH

parindent	12.0	pt	✓
hsize	15.0	cm	✓
baselineskip	12.0	pt	✓
lineskiplimit	1.0	pt	✓
lineskip	2.0	pt	✓
offinterlineskip	[ ]		✓

Figure 15: Choosing a different style file (`controls.sty`), and revealing a new set of style controls.

```
%------------------- controls -------------------
\usepackage[
           ]{controls}

\AtBeginDocument{
                \hyphenpenalty=50               %
                \exhyphenpenalty=50             %

                ...

              \delimitershortfall=5pt         %
               }

%------------------- hyperref -------------------
\usepackage[
            pdftex,                             %
            a4paper,                            %

            ...

            filebordercolor=blue,               %
           ]{hyperref}

%------------------- geometry -------------------
\usepackage[
            portrait,                           %
            a4paper,                            %
            textheight=20.0cm,                  %
            textwidth=10.0cm,                   %
           ]{geometry}
```

Figure 16: A typical config file.

Using the controls

The controls have been designed to be used intuitively and quickly. Pop-up menus in `choice`-type controls, for instance, are set using the mouse, while `number`-type controls can be set in a variety of ways: (1) direct input from the keyboard; (2) up and down arrows for increasing and decreasing the value; and (3) choosing a value from a small pop-up box. In cases (2) and (3), the change is determined by the `increment` value set in the data file.

The controls for each style file are shown on a separate page, or 'card', as each record is called in Revolution. By selecting the required style file from a pop-up menu on the left, the card corresponding to that style is shown. The main controls at the top remain, but the relevant style controls now appear below them, as can be seen in figure 15.

Writing the config file

When Batch Commander writes out the config file, it goes through each card (i.e. style file) in turn, gathers the data from the controls on that card, and then appends the data to the config file, to obtain a file as shown in figure 1(b). Figure 16 shows a typical config file. If `Immediate` is switched on, then as soon as any control is modified, the whole process of writing out the config file is redone, and the main file is run through TEX, in our case using TeXShop. This works quite well, and does not seem to slow down the interactive facility. If the user manipulates the controls before the end of the cycle of writing the file and running TEX, then the cycle is silently abandoned and restarted.

Status of Batch Commander

At the time of writing, the program is at an 'alpha' stage. When set up, it works well and with very fast feedback, but it needs work in several areas, in particular:

- Improving its general stability and reliability
- Some support for undoing actions
- Ability to read data from a config file and set controls according to it
- Support for flexible file management, directory structures, etc.
- Porting to different platforms, for which I will need the help of others.
- Improving the structure of the date file, in particular obviating the need for the '======', and allowing empty columns without a dash.

Availability

The program will be made available free of charge, although the license has not been finalized yet. If you would like to use the program, please mail the author.