ffslides: freeform slides and more based on the article class

Mark A. Wolters
November 19, 2015

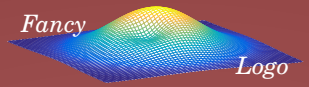
Abstract: The `ffslides` document class is intended for those who prefer a "do-it-yourself" approach to creating presentations. The distinguishing features of the class are:

1. Freedom to specify the page dimensions relative to the text size.
2. Freedom to design the elements of the slides (header, footer, background) as you like.
3. Freedom to place text, graphics, or annotations anywhere on the page you like, in any order you like.

The class is essentially a small set of macros added to the article class to make it easier to achieve these three goals. As a side benefit, the freeform nature of the class means it can also be used to produce posters, research notes, or other documents where one might not want a rigid, pre-specified visual layout.

This document was created with the class. Along with its LaTeX source, it can be used as both a reference manual and an example presentation.

Table of Contents



②

Should I Use this Class? **3**
Quick Start ③ **5**
Command Summary **6**
Essentials **7**
Setting the page size 8
Creating normal pages 9
Creating blank pages 10
Adding ctext and btext 11
Adding graphics 12
Customizing the Style **13**
Changing the background, header, and footer . 14
Styling dtext, ctext, and btext 15
Re-using your designs 16
Other Features **17**
Alignment grid 18
Hyperlinks 19
Quick items 20
Appendices **22**
tips and tricks 23
Internal commands and lengths 24

①

This document is created using the `ffslides` class. It is intended more to illustrate the capabilities of the class than as a model for creating a good design.

Throughout the document, boxes like this one will be used to provide comments and code examples elaborating on how content on the page was produced. For example:

① This box was created with command

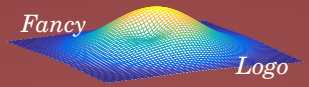
```
\btext[tr]{0.95}{0.13}{0.3}{%
    This document...
}
```

② The main text area on this page was set to take up only the left half of the page, using optional arguments to `\normalpage`:
`\normalpage[0.05][0.5]{Table of Contents}{...}`

③ The table of contents was generated by the usual `\tableofcontents` command. It was populated by adding the class's `\makesection` and `\makesubsection` commands on the appropriate slides.

The full source for this document is found in `ffslides-doc.tex`. An empty template can be found in `ffslides-template.tex`. The template has ample comments to guide the user through setting up their design.

Should I Use this Class?



The separation of content and format is one of the great strengths of TEX/LATEX. Nevertheless, presentations and posters are inherently visual and many of the strong points of TEX (automatic paragraph and page breaking, automatic placement of floats, etc.) are either irrelevant or limiting when creating them. A (probably) common experience is expending a great deal of effort to discover ways to override or circumvent the default decisions of the TEX system to make a slide look exactly the way you think it should.

There are of course a variety of classes already extant to make it more natural to produce presentations and/or posters. When using them, you may find that i) there are still some difficult-to-overcome default design choices that you might not like, or ii) there is a considerable learning curve with a large number of new commands.

The goal of this class is to provide a relatively small set of commands and a template that will allow the user to design a presentation from scratch to look the way they want; and further, to allow the user to place a variety of content freely on the page with minimal difficulty.

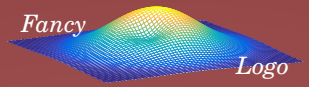
This was achieved by a simple strategy and heavy use of the package `pstricks`.

Here are the main ideas implemented in the class:

- The page height is user-specified in multiples of the line height, and the page's aspect ratio is also user-specified. The actual page dimensions are set using the `geometry` package.
- The user can define commands to draw the page background, header, and footer. These are automatically generated on each new slide.
- All content is placed on the page using `pstricks` macros (either directly, or through commands defined in the class). This means that everything on the page is considered by TEX to be zero width, and the TEX "current point" never moves from the upper left corner of the page. The upshot: there is a fixed coordinate system for everything on the page. Controlling the relative positions of different items is straightforward.
- The class is based on the article class, and makes no modifications to that class other than adding commands to achieve the aforementioned functionality. The class itself is pure LATEX, it doesn't use TEX commands.

This approach to creating presentations has a few consequences that will probably determine whether or not you will want to use the class. These are discussed on the next slide.

Should I Use this Class? (continued)



Consequences of the class's design:

1. If you want to achieve a certain effect in your presentation, you can probably do so, since the document is essentially an article with `pstricks` objects and `minipages` positioned here and there.
   - You can use your favorite packages and LaTeX tricks to achieve many custom effects.
   - You can use the extensive capabilities of `pstricks` as well as externally-produced graphics to customize the style to your heart's content.
   - On the other hand, you will need to know something about `pstricks`, how LaTeX handles boxes and space, how to define and redefine commands, and so on if you wish to do more than a basic presentation using the default style.
2. The number of new commands you will need to learn to use the class is fairly small.
   - The hope is that you will spend more time learning generally-useful LaTeX techniques than studying the manual of a presentation package.
3. There is no built-in functionality for fancy things like animated slides, speaker notes, and so on.
4. Using the package will necessarily involve tweaking the coordinates of the objects you place on the slides.
   - Depending on your patience level, an editor with a live/instant preview will be either a major bonus or a necessity.

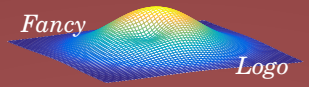
The next two slides give a **quick start** showing how to make a basic document, and a **command summary** listing all of the new commands in the class.

After you are familiar with the class these two pages can (hopefully) act as a reference guide while you're working.

The pages following the command summary explain how to use the class in more detail.

When creating your own presentations, it is recommended to start with the template **`ffslides-template.tex`**. The preamble of that document is structured and has lots of comments that describe what you should add to customize the presentation to your liking.

Quick Start



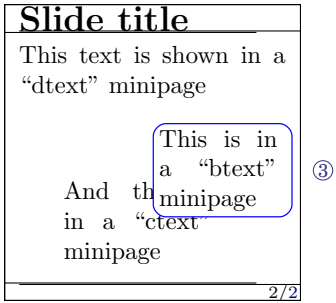
A minimal document. Here is the code for a document with two pages, and what the typeset pages look like.

```
\documentclass{ffslides}
① \ffpage{10}{1}
\begin{document}

\blankpage
\ctext[t]{0.5}{0.3}{0.5}{%
    \centering Blank page, title page}

\normalpage{Slide title}{%②
    This text is shown in a ''dtext'' minipage}
\ctext{0.2}{0.6}{0.5}{%
    And this is in a ''ctext'' minipage}
\btext{0.5}{0.4}{0.4}{%
    This is in a ''btext'' minipage}

\end{document}
```



Class options. Use `\documentclass[<options>]{ffslides}` to specify options. Available options are:

`showgrid`	Puts a 100×100 grid (with color `ffgridcolor`) over the background to help with alignment. See Alignment grid.
`<name>`	If a file named `bground-<name>.txt` is found, the code in that file is used to define the background. Similarly `header-<name>.txt` or `footer-<name>.txt` will be used to define the header and footer. This makes styles somewhat portable. See Changing the background...
others...	`draft, leqno, fleqn, openbib, 10pt, 11pt, 12pt` are passed on to the article class. Font size options are only included for potential font or package compatibility, since page size is set relative to line height. Other article-class options are not relevant.

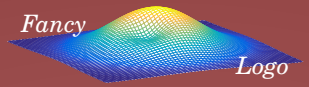
Notes

① `\ffpage` is used to set the page dimensions. In this example the page is 10 lines high and has aspect ratio 1 (it is square).

② Because all of the text is input as command arguments, including inadvertent white space can cause discrepancies in the layout. When in doubt, terminate lines with %.

③ Note that the `\btext` content sits on top of the `\ctext` content. Content is drawn in the order you enter it, which allows you to control overlap.

Command Summary



Coordinate system. The top left corner is (0,0) and the bottom right corner is (1,1).

Essential commands (creating pages and adding content). Here, `L`, `R`, `x`, and `width` are all fractions of the paper width, and `y` is a fraction of the paper height. `RP` is the `pstricks` the reference point of the box, e.g. `tl` (default) for top left, `r` for right center.

`\ffpage{height}{aspect}`	Use once in the preamble to set page dimensions.
`\normalpage[L][R]{title}{content}`	Create a new page with background, header, and footer.
`\blankpage`	Create a new page with only the background. Header or footer can be added with `\drawheader{title}` or `\drawfooter`.
`\ctext[RP]{x}{y}{width}{content}`	"Custom text:" place `content` in a minipage of the given `width` at (x,y).
`\btext[RP][style]{x}{y}{width}{content}`	"Boxed text:" just like `\ctext`, but the minipage is placed inside a framed box.
`\putfig[RP]{x}{y}{width}{file}`	Puts graphics `file` at (x,y).
`\dtext[L][R]{content}`	"Default text:" can be used to add the default text area to a `\blankpage`.

Customize the style (modify with `\renewcommand` in the preamble to change appearance)

`\drawbackground`	Used by `\normalpage` and `\blankpage` to draw the background.
`\drawheader{title}`	Used by `\normalpage` to create the header. Or to add one manually to blank pages.
`\drawfooter`	Used by `\normalpage` to create the footer. Or to add one manually to blank pages.
`\dtextleftedge`	Alias for a number in [0,1] specifying the left edge of the `\dtext` region.
`\dtextrightedge`	Alias for a number in [0,1] specifying the right edge of the `\dtext` region.
`\dtexttopedge`	Alias for a number in [0,1] specifying the top edge of the `\dtext` region.
`\dtextinclude`	Alias for commands to add just before `content` in every `\dtext` minipage.
`\ctextinclude`	Alias for commands to add just before `content` in every `\dtext` minipage.
`\btextinclude`	Alias for commands to add just before `content` in every `\btext` minipage.
`btextboxstyle`	Sets the default style of `\btext` boxes. Do not use `\renewcommand` for this one. Instead, use `\newpsstyle{btextboxstyle}{param_value_pairs}`.

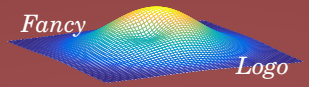
Convenience commands (not required, but could aid ease-of-use)

`\gridon, \gridoff`	Turn the alignment grid on or off at the next page. Overrides `showgrid` option.
`\makesection[targ_name]{toc_listing}`	Creates a phantom section for use in a table of contents. Optionally creates a hypertarget for internal linking as well.
`\makesubsection[targ_name]{toc_listing}`	Same as `\makesection`, but makes a phantom subsection instead.
`\qi[bullet]{item}`	Creates a "list-item-like" layout with `bullet` on the left and `item` on the right.
`\qitemi,\qitemii,\qitemiii`	Defines default `bullet` used by `\qi`, `\qii`, `\qiii`.
`\qii{item},\qiii{item}`	Alias for `\qi[\qitemii]{item}` and `\qi[\qitemiii]{item}`

Essentials

This section covers the main commands used to create pages (slides) and add content: `\ffpage`, `\normalpage`, `\blankpage`, `\ctext`, `\btext`, and `\putfig`.

Setting the page size



(0.05, 0.13)

To control the page dimensions, use `\ffpage{height}{aspect}`. This command should appear early in the preamble (right after `\documentclass` would make sense). It is a macro that uses the facilities of the `geometry` package to set the page dimensions.

Both `height` and `aspect` are numbers (not TEX lengths). `height` is the number of lines tall to make the page, and `aspect` is the aspect ratio (width/height) of the page. For example, the page size for this document was set using `\ffpage{35}{1.3333}`, which means:

- The page is 35 lines tall. The height of a line is measured by `\baselineskip`. The boxes at the left side of this page are all `\baselineskip` tall, and show that the page is in fact 35 lines high. For a real presentation a height of 25 lines or so is probably more suitable.
- The aspect ratio is 1.3333, yielding a 4:3 page.

To ensure that the origin of the coordinate system is at the top left corner of the page, the lengths `\topskip` and `\parindent` are both set to 0pt by the class. They can't be changed globally. Instead, if you want to change paragraph formatting or other aspects of the style of text in `\dtext`, `\ctext`, or `\btext` boxes, use the method described in the Styling dtext, ctext, and btext section.

(0.6, 0.5)

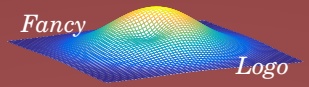
The **coordinate system** used to position material on the page is the same regardless of the page size. The top left corner is (0,0) and the bottom right corner is (1,1).

A few points are shown on this page with their coordinates, for illustration. From these points we can see:

- The top left corner of the `\dtext` region for this presentation is set to (0.05, 0.13).
- This page was created with the command `\normalpage[0.05][0.6]{Setting the page size}{To control the page dimensions...}`. This set the `\dtext` region of this page to occupy the horizontal part of the page from 0.05 to 0.6.
- The footer of this presentation is vertically positioned at the 0.975 point of the page.

(0.5, 0.975)

Creating normal pages



A "normal" page is one with a header, a footer, a background, and a default text area. Such a page is created with the command `\normalpage[L][R]{title}{content}`.

The command carries out the following operations:

- A `\newpage` is used to create a new page.
- `\drawbackground` is used to draw the background.
- `\drawheader{title}` is used to draw the header with the supplied `title`.
- `\drawfooter` is used to draw the footer.
- `\dtext[L][R]{content}` is used to put `content` on the page.

The `content` is placed in a minipage, so it can contain any LaTeX commands that will work in a minipage (that is, almost anything).

Optional arguments `L` and `R` are numbers between zero and one, giving the horizontal coordinates of the left and right edges of the text area.

Changing `L` and `R` makes it easy to make room on the slide for placing figures or other content. For this page, `L` was set to 0.15, and `R` was set to 0.7.

0.15

0.3

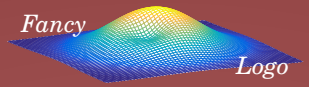
If the optional arguments are omitted, the text's left and right edges will be set to default values, which are held in the class-defined aliases `\dtextleftedge` and `\dtextrightedge`. The class sets these values to 0.05 and 0.95, respectively, but you can change the them in the preamble using `\renewcommand`.

It is possible to use `pstricks` commands directly to draw items on the page.

The red lines on this page were made with, e.g.
`\psline[linecolor=red]{<->}%`
`(0,0.7)(0.15,0.7)`

Any such commands will use the page's coordinate system if they are placed outside of a `\normalpage`, `\ctext`, or `\btext` command.

Creating blank pages



Blank pages, such as the main title page and section title pages in this document, can be created using `\blankpage`. This command takes no arguments.

The main purpose of `\blankpage` is for the creation of title pages. It could also be useful for special pages where the header and/or footer aren't desired, e.g. for pages with large images or tables.

All the command does is use `\newpage` to create a new page and `\drawbackground` to draw the background. If you want a different background for your blank pages you can draw it over top of the default background on a page-by-page basis.

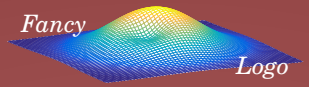
You may also add the elements you want manually after creating a blank page, using `\drawheader`, `\drawfooter`, and `\dtext`.

The following two code chunks will produce the same result:

```
\blankpage%
\drawheader{title}%
\drawfooter%
\dtext{content}
```

```
\normalpage{title}{content}
```

Adding \ctext and \btext



If you prefer to put all of your slides' content in a single minipage and use standard LaTeX methods of controlling where things go, you do not need to bother with the `\ctext` or `\btext` commands. When you use

```
\normalpage{title}{content}
```

to produce a page, the `content` is placed in a minipage. So you can place text, lists, tables, graphics, and so on as you wish in the normal way. You could create an entire presentation this way using only `\normalpage`.

The point of the `\ctext` ("custom text") and `\btext` ("boxed text") commands is to make it easier for you to place (possibly boxed) minipages at the location of your choosing. This is convenient once you start placing figures and annotations beside or on top of your main content. The syntax for these two commands is the same except `\btext` takes an extra optional argument:

```
\ctext[RP]{x}{y}{width}{content}
\btext[RP][style]{x}{y}{width}{content}
```

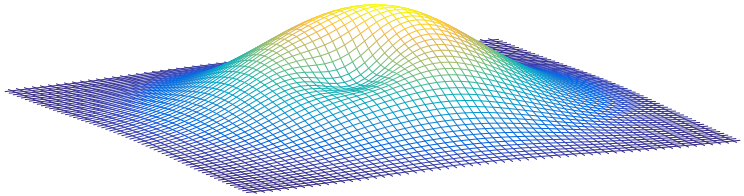
The required arguments `x`, `y`, and `width` are numbers between 0 and 1 that determine the minipage's placement and width. Optional argument `[RP]` is the reference point used by `pstricks` to place the box. It is a string of one or two characters: `t`, `b`, or `B` for vertical placement, `l` or `r` for horizontal placment. The default is `tl`, which places the box such that it's top left corner is at (`x`,`y`). See the `pstricks` documentation for more detail.

The optional argument `style` for `\btext` allows you to specify `pstricks`-style graphics parameters to control the box's style. A document-wide default style for the `\btext` boxes can be set in the preamble: see Styling dtext, ctext, and btext.

This red text was placed using `\ctext[bl]{0}{0.9}{0.4}{\red This red text...}`. The blue dot shows the location of the reference point.

This box was made using `\btext[t][linecolor=red,linearc=0,fillcolor=pink]{0.7}{0.8}{0.36}{This box was made...}`. The box follows the document-wide default style, except for the graphics parameters we've changed in the optional argument. The blue dot shows the reference point.

Adding graphics



A function that is unimodal, but still has a local minimum.

One option for adding graphics to the presentation is to use `\includegraphics` inside the `content` argument of a `\normalpage`, `\ctext`, or `\btext` command. This will put the graphic into the minipage in the usual way.

Another option is to use the command `\putfig[RP]{x}{y}{width}{file}`, where `file` is the path/name of a graphics file to include. The other arguments are exactly the same as those of `\ctext` and `\btext`. Using `\putfig` allows you to place the figure using the page's coordinate system.

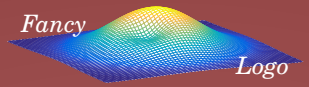
For example, on this page:

- `\normalpage` was used with optional arguments to place this main text region on the right half of the page.
- `\putfig` was used to locate the figure with the center of its top edge at (0.3, 0.35).
- `\ctext` was used to place the caption text with the center of its top edge at (0.3, 0.55).

Customizing the Style

The previous section described how to create pages and place content on a page. This section covers how to control the look of the presentation.

Changing the background, header, and footer



There is, of course, a price to be paid for the freedom to design the slides yourself: you have to have an idea what you want and the skills to produce it. In the `ffslides` class, the commands **`\drawbackground`**, **`\drawheader{title}`**, and **`\drawfooter`** are used to produce the background, header, and footer, respectively. You can redefine these in the preamble (using `\renewcommand`) to get the design you want.

① Please note:

- You can include arbitrary commands in any of these command definitions, including commands to draw `pstricks` objects, or `\ctext` and `\btext` commands to place text.
- The `\drawheader` command takes one argument, while `\drawbackground` and `\drawfooter` take none.
- On a normal page, the background is drawn first, then the header, then the footer.
- Any of these commands can place content anywhere on the page. The "fancy logo" in the upper right of this presentation, for example, could be drawn as part of the header, the background, or even the footer.
- The commands use the same coordinate system used throughout the class.

① The "Please note:" list was created using nested `\qi` quick items. See page 20.

For example placing the following definition

```
\renewcommand{\drawheader}[1]{%
  \ctext{0.05}{0.05}{0.9}{\LARGE\red Hello, World! #1}
  \pscircle(0.5,0.5){1em}}
```

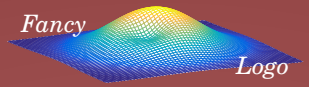
in the preamble would cause every page with a header to show Hello, World! before the page title, and to include a circle with radius 1em at the center of the page.

For drawing backgrounds and adding graphical flourishes to the design, there are essentially two approaches consistent with the design of the class:

1. Use the extensive capabilities of the `pstricks` packages to draw the design you want, or
2. Use an external program of your choice to create the background as a graphics file and add it to the design using `\includegraphics` or `\putfig`.

The `pstricks` packages are well documented and should be sufficient for most designs.

Styling \dtext, \ctext, and \btext



Changing \dtext positioning. If you create a standard page using \normalpage{title}{content}, your content is placed in a \dtext minipage with left, right, and top edges given by **\dtextleftedge**, **\dtextrightedge**, and **\dtexttopedge**. Each of these commands is an alias for a number between 0 and 1. You can redefine them in the preamble to change where the default text area is shown. For eample,

```
\renewcommand{dtexttopedge}{0.2}
```

will cause your text region's top edge to be 20% of the page height down from the top of the page. Tweaking the \dtext location like this is usually necessary to get your text area to look appropriate for your header and background.

Changing the default format of text. The commands **\dtextinclude**, **\ctextinclude**, and **\btextinclude** are placed just before the content of each of the corresponding minipages. By default each of these are empty, but you can use \renewcommand in the preamble to add any code you like. The main purpose of this is to let you set document-wide defaults for font size, color, and paragraph formatting in each type of box. For example, adding

```
\renewcommand{ctextinclude}{\small\red\setlength{\parskip}{1ex}}
```

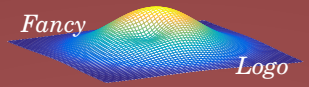
in the preamble will cause all \ctext minipages you create to have small red text and a \parskip of 1ex.

Changing the look of \btext boxes. The visual style of \btext boxes is set by parameter-value pairs ("graphics parameters" in pstricks parlance), and the defaults can't be set by \renewcommand. Instead we use pstricks's ability to save sets of graphics parameters to a named style. To change the default characteristics of all \btext boxes, include **\newpsstyle{btextboxstyle}{<param_value_pairs>}** in the preamble. For example,

```
\newpsstyle{btextboxstyle}{linecolor=red,fillstyle=solid,fillcolor=green}
```

would set all \btext boxes to have a red outline and green background by default.

Re-using your designs



The code for drawing the background, header, and footer can be located in external files, and added to the presentation using document class options. This gives slide designs a certain degree of portability and makes it easier to re-use your creations across documents.

To use this functionality, create some or all of the text files as shown here:

File name	File contents
`bground-<name>.txt`	`\newcommand{\drawbackground}{ <commands> }`
`header-<name>.txt`	`\newcommand{\drawheader}[1]{ <commands> }`
`footer-<name>.txt`	`\newcommand{\drawfooter}{ <commands> }`

Place the files alongside your source file so TEX can find them. Then specify the `<name>` part of the file names in the class options to control which files get used.

For example:

- `\documentclass[mystyle1]{ffslides}` Creates a document with background defined in `bground-mystyle1.txt`, header defined in `header-mystyle1.txt`, and footer defined in `footer-mystyle1.txt`, if those files can be found.
- `\documentclass[mystyle1,mystyle2]{ffslides}` Creates a document with background defined in `bground-mystyle1.txt`, and header/footer defined in `header-mystyle2.txt`, `footer-mystyle2.txt`, if those are the files present.

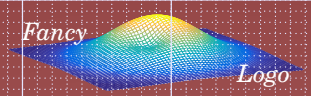
In this way the user can change change designs on the fly, or mix and match the background/headers/footers from different designs they may have. But the class will use the first background, header, or footer files it finds that match any optional argument(s), so the exact behaviour depends on which files are present.

These features are far from a full, robust "theming" system, but such functionality was not a goal of the class's design. The ability to store headers/footers/backgrounds externally was included as a convenience to the user who develops two or three favorite layouts they like to use for different purposes.

Other Features

This section describes a few features that have been included in the class purely for convenience—the class can be used perfectly well without them. It could be skipped on first reading, and in any case the choice to use these features or not is completely up to the user.

Alignment grid



If you choose to add annotations, figures, or objects to your slides using `\ctext`, `\btext`, `\putfig`, or the many `pstricks` commands, you will have to enter page coordinates to place the material. An alignment grid, like the one on this page, can speed up this process.

It is a 100×100 grid. In the class's coordinate system, (where the page is viewed as a unit square), the grid has major divisions every 0.1 and minor divisions every 0.01, in both directions.

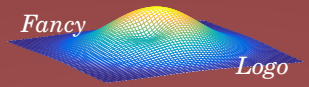
The grid can be turned on or off:

- Globally for the whole document, using the `showgrid` class option (include the option to show the grid on all pages; leave it out to not show the grid).
- Locally, using commands **`\gridon`** and **`\gridoff`**. These commands will control grid visibility on the *next* new page after their appearance, so they would normally be placed in between page creation commands. They take precedence over the class option.

The default color of the grid is a named color called `ffgridcolor`. The color can be redefined in the preamble, using `\newrgbcolor{ffgridcolor}{<R G B>}`.

The grid appears on top of the background, header, and footer, but underneath other page contents.

Sections and Hyperlinks



The usual LaTeX sectioning commands don't make a lot of sense in a presentation, where page breaking is done manually and different users might have different ideas about how to visually demarcate different "sections" of a talk. Hence the `ffslides` class is not designed with the use of sectioning commands in mind. In particular, users wanting "section numbers" in their slide titles will have to either enter them manually or devise their own way to use LaTeX counters.

Still, it is occasionally useful to include a table of contents (as in this document), and also nice to be able to provide PDF hyperlinks between sections. Two commands have been included in the class to provide these abilities:

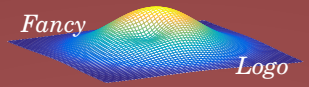
```
\makesection[targ_name]{toc_listing}
\makesubsection[targ_name]{toc_listing}
```

Either of these commands can be entered after a page creation command to achieve the following:

- Create a hypertarget with name `targ_name` that can be used to link to that page elsewhere (using the `\hyperlink` command)
- Create a phantom section or subsection (depending on which command was used) with name `toc_listing`. The phantom section won't change the document's appearance, but it will show up in the table of contents if the `\tableofcontents` command is used to produce one.

The hyperlinking functionality depends on the `hyperref` package, which is loaded automatically by the document class. See that package's documentation for how to use `\hypertarget` and `\hyperlink`.

\qi: quick item



The LaTeX list-making environments (`itemize`, `enumerate`, and `description`) are very useful, and can be customized as you like (especially if you use the `enumitem` package). Nevertheless, when preparing a presentation one may still wish for a simple structure to make "list-item-like" content layouts that allow the bullet/number/description and the spacing to be adjusted on an *ad hoc* basis as you go.

That is the purpose of the `\qi` macro. It creates an arrangement like the following (boxes added to aid visualization):

left side, typeset in "LR-mode."	**right side**, typeset in a minipage, top-aligned, with width equal to the remainder of the line width. It can contain anything that can go in a minipage, including additional nested `\qi` commands if one sees the need to use them.

This simple structure and the fact that you can nest commands means that you can quickly produce a variety of list-like or even table-like arrangements without needing other methods. The command's syntax is:

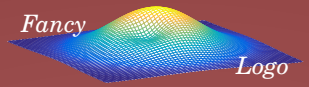
`\qi[<left_side>]{<right_side>}`	(the main command)
`\qii{<right_side>}`	(an alias for `\qitem[\qitemii]{<right_side>}`)
`\qiii{<right_side>}`	(an alias for `\qitem[\qitemiii]{<right_side>}`)

The optional `<left_side>` argument can be anything, but there are three pre-defined commands that you can use on the left side to produce something equivalent to an `enumerate` environment: `\qitemi`, `\qitemii`, and `\qitemiii`. You can redefine these in the preamble to set your own bullets. By default they are set to `\labelitemi`, `\labelitemii`, and `\labelitemiii`, to mimic the `enumerate` environment.

The default value of `<left_side>` is `\qitemi`. Commands `\qii` and `\qiii` are just shortcuts for making quick items with the default second- and third-level bullets.

Clearly, `\qi` can be used to quickly produce bulleted items without bothering to use `itemize` environments (or to set up their spacing and formatting). But other uses are possible. The next page gives some exaples.

\qi: quick item (continued)



Example 1

- We are producing a list
  - It has bullets at different levels
    - Its spacing is compact.
- Now it is over.

Example 2

α This example has an outer list with inter-item spacing equal to the document's paragraph spacing.

β It also has custom bullets
  $\Rightarrow$ With an inner list.
  $\Rightarrow$ Made using quick items nested inside the right side of item β.

γ Now it, too, is over.

Example 3

term One could use quick items for layouts like this.

word Of course `tabular` is the standard way, but with quick items we can build it line by line.

saying And we use a single command for everything.

The lists at left were made using the following code:

```
\textbf{Example 1}

\qi{We are producing a list}
\qii{It has bullets at different levels}
\qiii{Its spacing is compact.}
\qi{Now it is over.}

\textbf{Example 2}

\qi[$\alpha$~~]{This example has an outer list with inter-item
                spacing equal to the document's paragraph spacing.}

\qi[$\beta$~~]{It also has custom bullets\\
  \qi[$~~\Rightarrow$~]{With an inner list.}\\
  \qi[$~~\Rightarrow$~]{Made using quick items nested inside the
                        right side of item $\beta$.}}

\qi[$\gamma$~~]{Now it, too, is over.}

\textbf{Example 3}

\qi[\parbox{10ex}{~~\emph{term}}]{One could use quick items for
                                  layouts like this.}
\qi[\parbox{10ex}{~~\emph{word}}]{Of course \texttt{tabular} is
                          the standard way, but with quick items
                          we can build it line by line.}
\qi[\parbox{10ex}{~~\emph{saying}}]{And we use a single command
                                    for everything.}
```

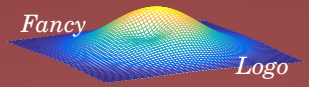
Appendices

Here we collect some additional information.

A1, Tips and tricks, gives a few usage tips. That slide is arranged to suggest how one could use the class to produce a research poster as a single-slide presentation with `\ctext` or `\btext` minipages arranged on the page.

A2 is included for the sake of completeness. It lists commands and lengths that are only used internally, by the class. It would only be of interest to those reading the class definition file.

A1: Tips and tricks



Creating a poster

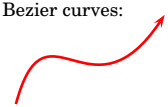
If we wanted to use `ffslides` to make a poster, we could set the page to have a large height relative to the text. here we've just used `\scriptsize` to shrink the text.

It's common to lay out posters as a sequence of panels. The class's `\btext` command is suitable for this. Inside each box we're working in a minipage, so we can use (almost) all of the usual LaTeX facilities.

It is possible to nest `\ctext`, `\btext`, `\qi`, and `\putfig` commands inside a `\btext`, so we can lay out a panel in a poster in a similar way to laying out a slide in a presentation. One caveat: the coordinate system changes when nesting. See the "nesting boxes" panel.

Direct use of `pstricks`

`pstricks` has commands for producing lines, curves, and shapes with control over line colors, fills, and much more. Some of these are useful for creating "callout" types of annotations on your slides. For example the `\psbezier` command produces Bezier curves:



`pstricks` commands `rput` and `uput` are also useful for placing objects and/or labels at specified locations.

Including `verbatim` text

Verbatim environments and the `\verb` command do not work when used in the arguments to a command. Since all page content in the `ffslides` class appears in command arguments, this is a problem.

There are various workarounds for this problem, that one can find by searching on the internet.

- Some suggestions may be found here: `http://www.tex.ac.uk/FAQ-verbwithin.html`
- see `\psverbboxtrue` and `\pslongbox` in. the pstricks documentation

The solution that was used in this document was to use package `fancyvrb`. This provides a `SaveVerbatim` environment that can be used (outside of any command arguments) to save verbatim code chunks to a name. Then this name can be used inside the command argument with, for example, the `\BUseVerbatim` command. See the `fancyvrb` documentation for more.

Nesting boxes

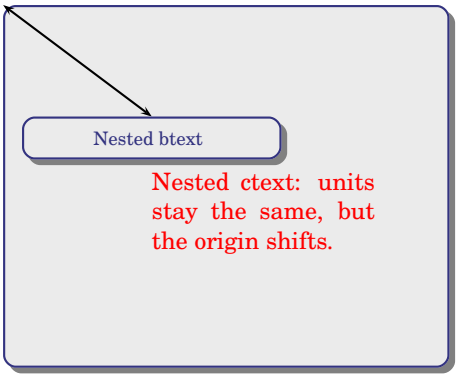
Normally in a presentation, `\btext`, `\ctext`, and `\putfig` are not put inside `content` argument of a `\normalpage`. This ensures that the coordinate system is the same for all items put on the page.

Nesting these commands is possible, however, and might be advantageous in certain situations, e.g.:

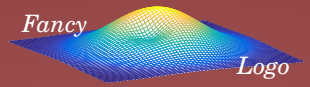
- When there is a graph or equation that has several annotations placed on top, nesting all of them inside a `\ctext` or `\btext` allows the whole group's location to be controlled by a single set of coordinates.
- When making a presentation, where everyhing in a panel is viewed as a group and might need to be moved around the poster together.

The only complication to remember is that once you are in a minipage, the TeX "current point" is no longer fixed at the top left corner of the page. So the $(0,0)$ point will be shifted to wherever the current point is. For example, the shaded box to the right was created with code:

```
\ctext[tl]{0.66}{0.595}{0}{%            %Make zero-width "container box"
  \psframe[style=btextboxstyle](0,0)(0.3,0.325)             %Draw a box
  \psline{<->}(0,0)(.1,.1)              %A line from (0,0) to (0.1,0.1)
  \btext[t]{0.1}{0.1}{.15}{\centering Nested btext}
  \ctext{0.1}{0.15}{0.15}{\small\red Nested ctext: units stay the same,
                          but the origin shifts.}}
```



A2: Internal Commands and Lengths



The following material could be an addendum to the command summary. It lists all of the internal commands and lengths that are used by the class, transparently to the user. Unless you are digging into the structure of the class definition `ffslides.cls`, there should be no need to know these items.

Internal commands (used by the class, not normally needed by the user)

`\showgrid`	Used to put the alignment grid on top of the background

Lengths (used by the class, not for direct modification by the user)

`\pgheight`	Used to set page height.
`\pgwidth`	Used to set page width.
`\dtextwidth`	Used in `\dtext`.
`\qitemwidth`	Used in `\qi`.
`\qbulletwidth`	Used in `\qi`.
`\labeliwidth`	Used in `\qi`.
`\labeliiwidth`	Used in `\qi`.
`\labeliiwidth`	Used in `\qi`.
`\labeliiiwidth`	Used in `\qi`.
`\qparskip`	Used in `\qi`.
`\outercapheight`	Used in `\qi`.
`\qgapheight`	Used in `\qi`.