

AcroT_EX Software Development Team

**The AcroT_EX Presentation Bundle
Manual of Usage**

**D. P. Story, Jürgen Gilg and
Simon Singer**

Table of Contents

1	Introduction	5
1.1	Original introduction	5
1.2	Introduction to APB 2.0	5
2	Software Notes	5
2.1	Software Requirements	6
2.2	Software Incompatibilities	6
3	Features of the AcroTeX Presentation Bundle	7
4	Some Comments on the OCG Approach	7
4.1	Advantages	7
4.2	Disadvantages	8
4.3	Sub-page Navigation	8
5	Getting Started with APB	9
5.1	Installation	10
	• Unzipping the APB Distribution	10
	• Installing the acrotex bundle	11
	• Installing apb.js	11
	• Erasing auxiliary files	11
5.2	Compiling your First Document	11
5.3	Using a Dvi-Previewer	13
5.4	APB Provided Sample & Tutorial Files	13
	• An APB Document Template	13
	• List of Tutorial Files	13
5.5	Options of APB and Web	14
	• Options of web	14
	• Options of APB	15
5.6	Font Sizes	16
5.7	Placement of Commands and Environments	17
5.8	The forscreen and forpaper Environments	17
6	The \maketitle and \tableofcontents Commands	18
6.1	The \maketitle Command	18
	• \DeclareDocInfo	18
	• The Title Page Structure	20
	• Greater Control of the Top Title Page	22
6.2	The \tableofcontents Command	24
7	Inserting Content with the slide Environment	25
7.1	The slide Environment	26
7.2	Simple Talking Points and Structured Lists	27
	• Simple Talking Points: \Bld	27
	• Structured Lists	28
7.3	Full Page Images	30
	• Moving the Panel	32
	• Full-Page Image and Printing	33

8 Navigating a Presentation	33
8.1 Sub-Page Navigation	33
• \ClearPagesOnClose and \DoNotClearPagesOnClose	33
• Sub-Page Navigation in a Browser	34
8.2 Navigation via Buttons	34
9 Customizing APB	34
9.1 Choosing the Screen Dimensions	34
9.2 Choosing between Full Width and Paneled Layouts	35
• Full Width Layout	35
• Paneled Layout	36
9.3 Choosing Background Color and Graphics	36
• General Concepts and Methods	36
• The APB themes	38
• Other APB Supplied Backgrounds	39
• User Defined Backgrounds and Colors	40
9.4 The Navigation Panel	40
• Top Panel: \insertLogo	40
• Middle Panel: paneloutline and \altmiddlepanel	40
• Bottom Panel and Navibar: The Navigation Buttons	41
9.5 Set Default Slide Options	43
9.6 Selecting Dings and Things	44
• Selecting Dings	44
• Selecting Colors	45
9.7 Section Headings	47
9.8 Running Headers and Footers	49
10 Refining the Presentation	49
10.1 Set Fullscreen Options	49
10.2 Set Page Transitions	51
10.3 Importing Sounds	52
10.4 The addJSToPageOpen/addJSToPageClose Environments	53
10.5 Special Acrobat Techniques	53
• Setting Transitions for Multiple Pages	53
• Locking Layers	54
• Auto Saving	54
11 Declaring the Initial View	55
12 Document Open Actions	56
13 Commands for Creating Layers	58
13.1 Simple OC	58
• \Bld	58
• \tBld	59
• \fBld	59
• \ocOff	61
13.2 Nested OCG	61
• \bBld	61
• \rBld	64
• \xBld	65
• \animeBld	66
• \stBld	69
13.3 Turning OC On and Off	71

14 Some Special Effects	71
14.1 Simple Special Effects	71
• Toggling and Dimming Color	71
• Toggling Text	72
14.2 Creating a Graphics/Captions Slide Show	73
14.3 The \slideshow Option	75
• A Graphics/Captions Slide Show Compatibility	76
• A Graphics/Captions Slide Show with Talking Points	76
14.4 The APB Help Feature	76
15 Printing the Presentation	76
15.1 Printable Copy of the Presentation	77
15.2 Thumbnails of the Presentation	78
• Create Thumbnails (no annots)	78
• Create Thumbnails (annots)	79
• Customizing the Thumbnail/Annot Generated File	83
16 APB and Exerquiz	84
16.1 Exerquiz Solutions	84
• The Title Pages for Solutions	85
• Multi-page Solutions	85
• The apbwriteto Environment	85
• Setting Slide Options	86
16.2 Stepping Through an Exercise or Quiz	87
• Exercises	87
• Forms in General	87
• Short Quizzes	88
• Quizzes	89
16.3 Crossing Slide Boundaries	90
• The exercise Environment	90
• The shortquiz Environment	91
• The quiz Environment	91
References	92

1. Introduction

Since its creation in 2007, APB has undergone one major revision (APB 2.0), with only a few minor revisions to reflect changes in dependent packages.

1.1. Original introduction

Well, here we are again, writing another \LaTeX package, this one for PDF presentations. The most significant feature that separates this package from the others— \TeX Power, Beamer, Prosper and the Utopia Presentation Bundle—is the use of **Optional Content Groups** (OCG), first introduced in **Acrobat 6.0 (Adobe Reader 6.0)**. OCGs will often be referred to more informally as *layers*.

As with any presentation package, there are two applications:

1. The package can be used to create “slides” for a talk in front of an audience;
2. The package can be used to write short presentations for classroom presentations or for distribution over the Internet.

This package keeps a very tight control over pages, *there is no flow of text from one page to another*. Content that flows over to the next page will be flagged, compilation will be stopped, and you will be asked to create an additional slide to handle the overflow. Therefore, a presentation—whether for an academic talk, the classroom, or distribution over the Internet—needs to be *designed with some thought*.

1.2. Introduction to APB 2.0

Version 2.0 of the [Acro \$\TeX\$ Presentation Bundle](#) introduces *sub-page navigation*, a major enhancement for navigating through an APB document while in *fullscreen mode*.

The notion of sub-page navigation was introduced in the *PDF Reference*, for PDF 1.5, which corresponds to **Acrobat/Adobe Reader** version 6. With sub-page navigation, you can navigate not only from page to page, but from layer to layer on the same page.

In the previous, and first version of APB, an elaborate system of buttons was created to navigate through the presentation, between layers and between pages. The presenter had to have cursor control at all times to click on buttons to advance the presentation. Remote, wireless devices—popular with PowerPoint presentations—*could not really be used* due to the need for mouse control.

In this latest release of APB, there is an option, `spnavi`, which causes APB to use sub-page navigation. Sub-page navigation comes into effect only in fullscreen mode. When sub-page navigation is available and the viewer application is in fullscreen mode, the presentation can be navigated through with the usual controls: left and right arrows, or left and right mouse clicks; consequently, an APB document *now can be controlled by remote wireless presentation devices*.

In addition to the `spnavi` option, there have been modifications of the standard build commands, some optional parameters have been changed; overall, the build commands have enhanced optional parameters to control its initial state, whether the layer should be printable (even if the layer is hidden), whether a sub-page navigation node should be created for a particular layer. There are optional parameters for creating transition effects for a layer and for executing JavaScript actions when a layer is made visible or hidden. These changes and enhancements are duly noted in the documentation.

2. Software Notes

In this section we discuss software requirements, and the incompatibilities this package has with the newly released [AeB Pro](#).

2.1. Software Requirements

First things first: The major requirement of this package is **Acrobat Pro 7.0** or later;¹ to repeat

Acrobat Pro 7.0, or later and accompanying **Distiller**

are required for this package to perform as designed. Once the presentation is built, however, **Adobe Reader 7.0** or later is sufficient to view the document.

This is a reasonable restriction as neither `pdftex` nor `dvipdfm`, the two major applications used by most of the \TeX -users to produce PDF (along with `pdfwrite`²), can produce the markup for Optional Content Groups. Therefore, I assume you are using **Acrobat Pro 7.0** and the accompanying **Distiller**. This package supports the use of `dvips` and `dvipsone` to produce a postscript file to distill.


The **Acro \TeX Presentation Bundle** (APB) requires the `web` package and the `eforms` package, both of which are included with the **Acro \TeX eEducation Bundle** (AeB) distribution. Below is a list of other required packages used by the APB:

1. `hyperref`: The `hyperref` bundle should be already on your system, it is standard to most \TeX distributions.
2. `xkeyval`: The very excellent package by Hendri Adriaens. This package allows developers to write commands that take a variety of complex optional arguments. You should get the most recent version, at this writing, the latest is v2.5e (2005/11/25) or later.
3. `xcolor`: An amazing color package by Dr. Uwe Kern. This package makes it easy to write commands to dim the color. Get a recent version, at this writing, the latest is v2.08 (2005/11/25).
4. `truncate`: This package, by Donald Arseneau, is used in the navigation panel to abbreviate the section titles if they are too wide for the panel. This package is distributed with the APB.
5. `comment`: A general purpose package, Victor Eijkhout, for creating environments that can be included in the document or excluded as comments. A very useful package for \TeX package developers. This package is distributed with the APB.
6. `eso-pic` by Rolf Niepraschk and `everyshi` by Martin Schröder, these are used by `web` to create background graphics and graphic overlays.

☛ One of the extremely nice features of **MiK \TeX** is that it can automatically download and install any unknown packages onto your hard drive, so getting the APB up and running is not a problem!

Some of the example files use other packages, in particular `PSTricks` and `fp`. These can be downloaded as needed.

2.2. Software Incompatibilities

Concurrent with the release of APB 2.0, comes the release of another package termed **AeB Pro**. These two packages share many common features; in fact, several useful features of **AeB Pro** were transferred to APB, and several features of APB were incorporated into **AeB Pro**. That having been said, the two packages are **incompatible**. Do not try to load **AeB Pro** either before or after loading APB. The two simply do not coexist with each other! So, *don't try it!* 

¹In the United States and Europe, Adobe offers a significant academic discount on its software, including **Acrobat**. Educators should look into the price structure of **Adobe Acrobat** at their institutions; perhaps, their Department or College can supply a financial grant for the purchase of the software.

²I know very little of `pdfwrite` and its capabilities.

3. Features of the AcroT_EX Presentation Bundle

The AcroT_EX Presentation Bundle has many features, some are standard to L^AT_EX Presentation packages, others are new to APB. Below, we list some of the major features:

1. Sub-page navigation, so the presentation can be navigated using the standard set of controls.
2. Animation: Using the layer feature of PDF, it is possible to create interesting animations.
3. APB offers a library of graphical backgrounds and buttons. There is also a collection of professionally designed “theme” backgrounds ready for use “out of the box”. There is an easy to use system of introducing the graphical backgrounds, removing them, and changing them. This gives you many, many options for designing the visual look of your presentation.
4. A standard feature of presentation packages is support for PDF page transitions, and APB is no different. With APB, you have control over your full screen preferences and the full range of page transitions.
5. Support for importing sound clips and for playing clips on change of layer or page.
6. Support for creating slide shows.
7. Support for a toggle sideshow, which can be used to create a thermometer to show progress through the document.
8. The ability to easily customize the look of section, subsection and subsubsection headings.
9. There are several ways of making a printable version of the presentation.
 - (a) There is a `forpaper` option to create a paper version of the presentation; for details, see [Section 15.1](#).
 - (b) **Acrobat** menu control for creating a two-up thumbnail document of the talk. See ‘[Create Thumbnails \(no annots\)](#)’ on page 78.
 - (c) With APB, slides can be annotated, commented on. These comments, along with thumbnail images of each slide can be placed in a programmatically generated file. There are a number of options for the creation of these “thumbnail/annot” files. See ‘[Create Thumbnails \(annots\)](#)’ on page 79 for details.
10. Special effects: toggling, highlighting, dimming, stepping.
11. Graphic show (toggling through pictures very easily).
12. Compatibility with the AcroT_EX eEducation Bundle, in particular, exerquiz quizzes and exercises can be layered.

4. Some Comments on the OCG Approach

In this section we discuss some pluses and minuses of the use of Optional Content Groups.

4.1. Advantages

The packages mentioned earlier obtained their special effects by producing a large number of pages, with incremental content on each. As the user paged through the document, the viewer is left with the impression that additional talking points are appearing on the screen on that same page. This idea is quite workable, but produces a large number of pages for a typical presentation and there is not a one-to-one correspondence between slides and pages. Writing a presentation package for L^AT_EX is quite a task, the problem of incrementally building pages requires some clever T_EX/L^AT_EX programming skills.

The use of layers simplifies the job of writing a presentation package, but at the expense of other technical problems that needed to be identified, solved and overcome. One immediate advantage is

there is now a one-to-one relation between slides and pages! Though I haven't made any direct comparisons, I'm relatively confident that the file size of a presentation created by APB will be smaller than the same presentation built by one of the other packages.

With the release of APB 2.0, the control of an APB document becomes more like that of a PowerPoint document. You can now progress through your presentation exactly the way you would with PowerPoint, using the standard navigation controls of left/right arrow keys or mouse clicks. Wireless presentation devices can now be used to navigate through an APB presentation.

4.2. Disadvantages

One such problem is the problem of “paging” through the presentation. For traditional packages, we simply page through the document. In the case of OCG, we are doing a combination of paging and stepping through the layers. As a result, ‘intelligent’ JavaScript has been written that allows us to step through the talk by pressing a Next button. The JavaScript is intelligent in the sense that it can detect a page boundary, and automatically increment the page. The JavaScript also detects the code marked for dimming, toggling, or other “specials” and steps through these layers, a little differently.

There are several other very noteworthy negatives:

1. **Screen Flickering:** When the viewer is in full screen mode—the mode most talks are given in, depending on your computer monitor, there can be a noticeable screen flicker as layers of content become visible. This is not so pronounced with a plain white or colored background, but can be rather severe if there is a graphical background.

Reducing the screen resolution helps the problem. My monitor is normally set on 1600×1200 , when I reduce the resolution to 1024×768 there was virtually no screen flicker, that's good.

My LCD laptop, on which I give my presentations, performs pretty well; the screen resolution for the laptop is 1024×768 .


2. With heavy graphical content, the screen may be slow to refresh.
3. Performance of OCG animations decreases when there are one or more graphical backgrounds, so I recommend that a plain background be used on pages with OCG animations. Commands are provided that help you easily change, or remove backgrounds, so this should not be a problem.
4. An advantage of traditional methods of paging through the presentation is that transition effects can be used to display an additional point, because that point appears on the next page and transitions are a phenomena of pagination. In the case of layers, we cannot produce transition effects as we make a layer visible. The APB does support transitions between pages, however.

I have brought these problems to the attention of the OCG development team at Adobe and I am keeping my fingers crossed that when **Acrobat/Reader 8.0** comes out, many of these problems will be solved, the screen flicker and the slow performance of OCG animation over a graphical background, for example. I have requested that transition effects can be initiated programmatically with JavaScript; when this is the case, one can create a transition effect between layers, because you can associate a JavaScript action with a layer.

If these problems are solved by Adobe, this may be a rather popular presentation package.

4.3. Sub-page Navigation

APB 2.0 introduces the notion of sub-page navigation, which represents a big advance forward in the control of an APB document. The following paragraphs in this section can be skipped over at first reading. Come back to them at a later time, when you need more technical information about sub-page navigation.

-  Sub-page navigation is discussed in section 8.3.3 of the *PDF Reference* [4]. This section generally discusses presentations and page transition effects. The document `apb_snavi` contains this content, and an interactive demonstration of sub-page navigation.

To use sub-page navigation, the application must create a doubly-linked list of navigation nodes. The root (or primary) node is placed in the page dictionary. Each navigation node is a dictionary (PDF terminology) consisting of five (optional) key-values. These five controls are:

1. **NA:** (next action) A sequence of actions executed when the user navigates forward from this node.
2. **PA:** (previous action) A sequence of actions executed when the user navigates backward from this node.
3. **Next:** A pointer to the next node, if any.
4. **Prev:** A pointer to the previous node, if any.
5. **Dur:** The maximum number of seconds before the viewer application should advance forward to the next navigation node. If there is no **Dur** entry, automatic navigation does not occur.

The *PDF Reference* introduces the notion of a *current node*. The following describes the events associated with a request to navigate forward or backward.

- Navigate forward
 1. The sequence of actions as specified by the **NA** key, if present, is executed.
 2. The node specified by the **Next** key, if present, becomes the new current node.
- Navigate backward
 1. The sequence of actions as specified by the **PA** key, if present, is executed.
 2. The node specified by the **Prev** key, if present, becomes the new current node.

The *PDF Reference* notes that when navigating between nodes, it is possible to have *transition effects*. This is a very exciting feature that APB supports.

When there is a request to navigate to another page, the following events occur:

1. The root navigation node, if there is one, becomes the current node for the page.
2. If the navigation request was forward or if the navigation request was by a link to another page, the actions of the **NA** key of the root node are executed, and the current node becomes the one pointed to by the **Next** key.
If the navigation request is backward, the actions specified by the **PA** key are executed and the node specified by **Prev** becomes the current node.
3. The viewer application makes a new page, and displays it, using a transition effect, if any is specified.

Most of the standard build commands have new optional parameters for setting the values of **NA**, **PA** and **Dur** through a xkeyval interface using the keys `nextaction`, `prevaction` and `dur`. There are two additional keys for associating transition effects with a layer, these are `nexttrans` and `prevtrans`. These key-value pairs are described in [Table 1](#), page 10, for reference when the individual build commands are described.

Recognized string values for keys `nexttrans` and `prevtrans` ([Table 1](#)) are the same as the values of the `Trans` key of `\setDefaultFS`. See [Section 10.1](#), page 49.

5. Getting Started with APB

In this section, we try to explain the basics of using the APB. It is important to note that the `web` package is required, and APB works closely with `web`. Many options are controlled through the options of the `web` package.

We begin with the installation procedure. To get the APB system up and running, you must install both the APB distribution and the AeB. Follow the installation instructions of the next section closely.

Key	Value	Description
<code>nextaction</code>	JS	The JavaScript to be executed when the user requests forward navigation from the current node. The default action is <code>\setLayerNext</code> , which shows the associated layer. If custom actions are needed, use the <code>nextaction</code> key, but include <code>\setLayerNext</code> .
<code>prevaction</code>	JS	The JavaScript executed when the user requests backward navigation from the current node. The default action is <code>\setLayerPrev</code> , which hides the associated layer. If a custom action is needed, include <code>\setLayerPrev</code> as part of the action.
<code>dur</code>	Number	The maximum number of seconds before the viewer automatically should advance forward to the next navigation node. If this key is not present, no automatic advance occurs. The duration key is local to the build command that creates it; there is a global duration command, <code>\spNaviDuration</code> . By saying <code>\spNaviDuration{5}</code> , navigation automatically advances forward through all nodes, unless the command <code>\spNaviDurationClear</code> is executed.
<code>nexttrans</code>	String	The transition effects for the current node when forward navigation is requested. The value of this key is one of the recognized strings. For example, <code>nexttrans={FlyInDown}</code> .
<code>prevtrans</code>	String	The transition effects for the current node, when backward navigation is requested. The value of this key is one of the recognized strings. For example, <code>nexttrans={Fade}</code> .

Table 1: Key-value Pairs for Sub-page Navigation

5.1. Installation

This section contains the instructions for installing the APB so that it will perform as designed. The installation procedure described here includes: the unzipping and the placement of the `apb.zip` distribution; the placement of the `apb.js` JavaScript file to add some additional functionality to **Adobe Acrobat Pro 7.0** (or later); for users of WinEdt, the installation of a file to erase working files created by APB and AeB.

• Unzipping the APB Distribution

The APB distribution comes in the form of a single file, `apb.zip` with an approximate size of 20MB.

- If you have MiKTeX 2.8 or later,³ you need to install all packages by hand in a local TDS tree that you create. Review the MiKTeX help page on this topic,

<http://docs.miktex.org/manual/localadditions.html>

Within the `C:\Local TeX Files\tex\latex` folder,⁴ copy any ZIP files with the APB distribution, and unzip them. This creates an `apb` folder. Now follow the instructions below.

- After unzipping `apb` into the `apb` folder of your root folder

Users of **MiKTeX** or **T_EX Live** need to refresh the filename database.

After you've refreshed your filename database, the package files are ready to be used, but wait, APB requires the AeB (AcroT_EX eDucation Bundle, or just `acrotex`) and the installation of the `apb.js` file. These additional steps are described below.

³Latest version is MiKTeX 2.9

⁴The suggested name of the local folder, but you can choose any name and most any path.

- **Installing the acrotex bundle**

The web, eforms and insdljs packages of [AeB](#) are *required* by APB. These packages are part of the [AcroTeX eEducation Bundle](#) (the acrotex bundle). Use your TeX system's (MiKTeX or TeX Live) *download manager* to install the acrotex bundle.

Installation of aeb.js (Important) The aeb.js JavaScript file is required for users of the 'dvips + Acrobat Distiller' workflow. Carefully read the installation instructions given in `install_jsfiles.pdf` that is found in the doc folder of the acrotex installation.

- **Installing apb.js**

The JavaScript file apb.js that comes with APB creates a sub-menu entry, 'AcroTeX Presentation Bundle', under the View menu of the Acrobat toolbar. (**Note:** For Acrobat 9 or prior, this menu is under the Tools menu.)

The View > AcroTeX Presentation Bundle menu (for Acrobat X or later) is used to create thumbnails of your presentation. Use of this menu is discussed in [Section 15](#) on page 76.

To install apb.js (Important) Follow the instructions given in `install_jsfiles.pdf`, a document found in the folder apb/doc.

- **Erasing auxiliary files**

AeB and the APB create a number of auxiliary files with a variety of extensions: .cut, .sol, .qsl, .fdf, etc. These may be deleted after the PDF has been built.

For the WinEdt text editor. We provide `Erase Working Files.edt`, found in the apb/extras folder, to erase auxiliary file generated by AeB and APB.

Install this file as follows:

1. Go to the folder

`C:\Users\(username)\AppData\Roaming\WinEdt Team\WinEdt 10\Exec`

and copy the APB provided file `Erase Working Files.edt` into the folder at the end of the above path.

2. If you had already created a customized `Erase Working Files.edt` already, be sure to copy and paste your own special `AddFileItem` commands into the `Erase Working Files.edt` file.

The `Erase Working Files.edt` is executed by pressing the 'Erase Output Files' button (the one having a trash can as its icon) on the WinEdt toolbar.

For other TeX editors. If you use some other editor for composing TeX, perhaps it has an erase or clean feature. The following extensions can be erased after you've build an APB document: *.sol, *.qsl, *.djs, *.fdf, *.ljs, *.cut.

5.2. Compiling your First Document

There are three important rules for compiling an APB document:

- **Compile Three Times:** An APB document needs to be *compiled three times* before converting to postscript and distillation.
- **All Files in Same Folder:** Compiling creates certain .fdf files that are deposited in the folder of your source file. These .fdf files must be in the same folder as your .ps postscript file as you distill. The newly created .pdf file should also be saved into the same folder that contain these .fdf files.
- **Save:** After the PDF document is first opened in **Acrobat** following distillation, *save the document*. (When the PDF is opened for the first time after distillation, **Acrobat** will look for the .fdf files created during the TeX compile step. Saving the PDF will save the JavaScript code contained in these .fdf files in your document.) After saving, the PDF no longer needs the .fdf files and can be safely moved to another folder or the Internet.

Section 5: Getting Started with APB

ex1 **Example 1:** The following is a minimal example, it is somewhat plain, but it is a first start. See the file `apb/doc/tutorial/apb_ex1.tex`, compile three times, convert to a PostScript file using `dvips` or `dvipsone` and `distill` to produce the PDF.

If you compile this sample file to create `apb_ex1.pdf`, after the file appears on the viewer, click on the 'Next' button. The 'Next' button (▶) is the sixth button from the left in the row of navigation icons at the bottom of the page. Clicking on '▶' moves to the next page. You should see the section heading 'My First Section'. Click on '▶' again, at which point the phrase 'Welcome to Planet AcroTeX!' should appear. [Figure 1](#) on page 12 shows a verbose listing of `apb_ex1.tex` as well as images of the pages of the PDF.

```
1 \documentclass{article}
2 \usepackage[
3     dvips,          %<-- dvips or dvipsone
4     designv,
5     navbar,
6     nodirectory
7 ]{web}
8 \usepackage{apb}
9
10 \DeclareDocInfo
11 {
12     title=My First Presentation,
13     author=D. P. Speaker,
14     university=My University,
15     email=dpspeaker@mku.edu,
16     talkdate={Dec.\ 17, \the\year},
17     talksite=The Talking University,
18     subject=A basic APB template,
19 }
20
21 \begin{document}
22
23 \begin{slide}
24 \maketitle
25 \end{slide}
26
27 \begin{slide}
28 \section{My First Section}
29 \Bld Welcome to Planet AcroTeX!
30 \end{slide}
31
32 \end{document}
```

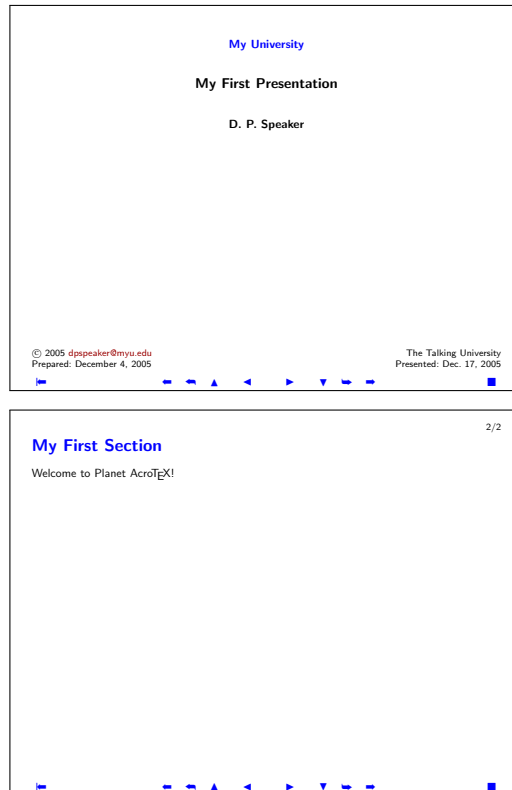


Figure 1: First File: `apb_ex1.pdf`

The figures on the right in [Figure 1](#) show the rather plain-looking PDF created by this listing. It shows, on the first page, the location of the document information introduced in lines (10)–(15), and it shows the navigation bar, the Navibar, running along the bottom of the two pages.

Comments: (1) We use the `article` class, but any class that has a `\section`, `\subsection`, etc., command structure will work; (2)–(7) We introduce the `web` package, the driver `dvips`, the design choice, `designv`, the `navbar` option so we get navigation buttons (important), and other options; (8) We introduce the APB package with no options (See [Section 5.5](#) for details on the options of the `web` and APB packages); (10)–(19) We enter some information about the presentation, this information will be used on the title page, see page 18, on '`\DeclareDocInfo`', for details.

We now come to the body of the document.

- **All content must be in a slide environment**, including the `\maketitle`, see lines (23)–(25). The `\tableofcontents`, which is not used in this document must be enclosed in a `slide` environment as well.

The second slide, (27)–(30), contains the first page of the talk. In line (28) we have the standard `\section` command, and in line (29) we have our content. Note line (29), the `\Bld` command creates a hidden layer, when the user clicks on the ‘Next’ button, the line ‘Welcome to Planet AcroTeX!’ will appear.

5.3. Using a Dvi-Previewer

The previewer Yap that comes with MiKTeX has its problems. Sometimes, Yap does not handle well Postscript specials that it does not understand; often no preview is given or Yap crashes. It is recommended, therefore, that the Default render method is set on Dvips. To set the default render method, open Yap and select View > Options from the menu bar; choose the Display tab and select Dvips from the dropdown menu labeled Default render method.

5.4. APB Provided Sample & Tutorial Files

The APB comes with several sets of files to help you get started with the APB: A document skeleton (or template) that is used to begin writing your own presentation; and a series of tutorial files to help familiarize yourself with some of the major features of APB

- **An APB Document Template**

The APB distribution comes with two template files; they are

- `apb_template.tex`: A template for building your own presentation, most all commands for designing your document are included. Make a copy of this file, and change it as desired.
- `apb_template_ud.def`: This file shows how to introduce user defined backgrounds, and lists various commands for selecting colors for the background.

These files are located in the `src_template` folder.

- **List of Tutorial Files**

Throughout this document, some examples are snippets of working examples, which can be regarded as a series of tutorials to discover the features of APB. These files are located in the `apb/doc/tutorial`.

- The name of each file has the form `apb_ex⟨n⟩.tex`. A box `ex⟨n⟩` appears in the left-margin of any example that uses code from one of these tutorial files. (A link is set to the `.tex` file as a convenient way of loading it into your favorite TeX editor.)

The listing of the working files follows, each containing a brief description and a reference to the example in which they are first discussed:

- `apb_ex1`: A first APB file; the starting point of the tutorial. See [Example 1](#), page 12.
- `apb_ex2`: This file illustrates the simple `\Bld` and `\ocOff` commands for creating talking points. See [Example 10](#), page 27 and [Example 11](#), page 27.
- `apb_ex3`: This file illustrates the list structure and dimming capability of APB. Refer to [Example 12](#), page 28; [Example 13](#), page 29; [Example 14](#), page 30.
- `apb_ex4`: This file teaches the basics of inserting background colors, background graphics, insertion of a watermark, removing background graphics, changing background colors and graphics, removing the watermark, and inserting it again. See [Example 21](#) and [Example 22](#), beginning on page 37.
- `apb_ex5`: This file shows of the nine themes distributed with the APB product; template management, creating a slide show, turning off layers, moving the navigation panel from right to left, or removing it entirely. See [Example 25](#), page 39.
- `apb_ex6`: This file demonstrates the thirteen backgrounds supplied by APB. It also demonstrates the use of `\useApbBg`. See examples [Example 26](#) and [Example 27](#) on page 39.

Section 5: Getting Started with APB

- `apb_ex7`: Demonstrates the `\sectionLayout` command for designing section, subsection and subsubsection titles. See example [Example 36](#) on page 48.
- `apb_ex8`: Illustrates ‘[Toggling and Dimming Color](#)’ on page 71, see, [Examples 55, 56, 57 and 58](#).
- `apb_ex9`: This file contains examples illustrating how to create graphics/captions slide shows (‘[Creating a Graphics/Captions Slide Show](#)’ on page 73). See [Examples 59 and 60](#) on pages 75 and 75, respectively.
- `apb_ex10`: This file shows how to create and manage a document compiled with the `slideshow` option. See [Example 61](#) on page 76.
- `apb_ex11`: This file shows how to create a document slide show that consists of only graphics/captions slides with no “talking points”. See [Example 62](#) on page 76.
- `apb_ex12`: This file shows how to create a document slide show that consists of only graphics/captions slides with “talking points”. See [Example 63](#) on page 76.
- `apb_ex13`: Contains the anime examples discussed in [Examples 52 and 53](#), pages 67 and 69, respectively.
- `apb_ex14`: Demonstrates how to remove templates, and how to restore them again. The file has some discussion on how graphics effect the performance of animations. See [Example 23](#) on page 38.
- `apb_ex15`: Discusses how to develop an APB document for both a presentation and for print. See [Example 64](#) on page 77.
- `apb_ex16`: This file is a tutorial on creating a thumbnail file with annotations of an APB document. See [Example 65](#) on page 79.
- `apb_ex17`: This file illustrates the design structure of the title page, and demonstrates how the commands `\universityLayout`, `\titleLayout` and `\authorLayout` are used to change the title format. See [Example 5](#) on page 23.

The following tutorial files are new to APB 2.0 and are meant to illustrate new features, such as sub-page navigation.

- `apb_ex18`: An introduction to the new APB 2.0 feature of sub-page navigation.
- `apb_ex19`: This is `apb_ex3` (`\dPt`, `\ddPt`, etc.) reworked a little to use sub-page navigation. It also illustrates a new APB 2.0 feature for creating enumerated lists.
- `apb_ex20`: (APB 2.0) Same as `apb_ex8` ([Toggling and Dimming Color](#)), but modified for use with the `snavi` option.

5.5. Options of APB and Web

By specifying various combinations of options in the APB and web packages, you can add features or change the appearance of the presentation document.

• Options of web

The web Package has a number of options, we mention only the ones that are useful in conjunction with APB. See the complete documentation of the web package, contained in the [AcroTeX Documentation](#).

The following are the options of the web Package.

- `dvips`: Specifies the dvi-postscript converter.
- `tight`: When this option is used, many of the list parameters are redefined so that there is not so much space around these environments, and between items.

Section 5: Getting Started with APB

- **nodirectory**: The directory automatically appears on the first page, which gives links to the table of contents and the beginning of the article. The directory will not appear if this option is used. Normally, you'd use this option with APB.
- **navibar**: Use the `navibar` option of `web` to add a navigation toolbar (referred to as the Navibar), as seen at the bottom of this page. This option is not needed if one of the panel options is taken.
The Navibar can be turned off and on from within the document by using the web commands `\NaviBarOn` and `\NaviBarOff`.
- **usetemplates**: The `usetemplates` option activates the mechanism for creating colored backgrounds and graphic overlays. When the `navibar` option is taken, navigation icons appear at the bottom of the page.
- **rightpanel**, **leftpanel**: When either of these two options is specified, a vertical panel is created.
- **designi-designvi**: Six options to choose the size of your presentation page. `designv` is the one that seems best suited for a presentation. Custom page sizes can be specified using the `\margins` and `\screensize` from the `web` Package.
- **draft**: When this option is taken, graphic overlays are not allowed. This is useful when you rely heavily on graphic overlays, but during the development phase, don't need to read and re-read your overlays. The defined background colors will be used instead. Remove this option to build the final version of your document.
- **forpaper**, **forcolorpaper**: The `forpaper` option is used to remove the color from the document, and to restore the standard `\textheight` of a standard `article` class \LaTeX document. The `\textwidth` is determined by the `\screensize` and `\margins` parameters or by the `design` option. The `forcolorpaper` does the same as `forpaper`, except it does not remove the color.
- **dvipsnames**: Any unknown options are passed to the color package (either `xcolor` or `color`). The option `dvipsnames` is one such example. The `dvipsnames` option causes the color package to read a list of definitions of many different colors. This may be handy in creating a colorful presentation.

- **Options of APB**

The following are the options of the [AcroTeX Presentation Bundle](#).

- (APB v2.0) **spnavi**: Option to include the necessary PostScript code for sub-page navigation. See the section on Sub-page Navigation, [page 8](#), for a description.
- (APB v2.0) **spnavibar**: Use the sub-page navigation bar, this is an abbreviated version of the navigation bar of `web`. When using sub-page navigation, a large number of controls are not needed.
- (APB v2.0) **sptrace**: A tool used in the development of sub-page navigation. When option is taken, messages are written to the console window as you navigate through the sub-page navigation doubly-linked list of nodes.
- **dim**, **dimUp** and **nodimming**: Global options to turn on dimming. We can dim down (`dim`) or dim up (`dimUp`). The `nodimming` option is a global override of the other dimming options and local commands. Below are defined additional commands `\DimOn` and `\DimUp` for locally changing the dimming of the content. The `nodimming` option will override these if there are any dispersed throughout the document.
- **draft**: This option passes the `draft` option to the `graphicx` package. It redefines `\Gin@setfile` of the graphic bundle so that the file names are not shown in the bounding boxes. Useful for previewing the layout of your talk, without including all the graphics. To see the file names use `\showfilenames>true` in the preamble of your document.

Section 5: Getting Started with APB

- **theme:** Use the `theme` key to use one of the themes provided by the APB. Recognized values are `mercury`, `venus`, `earth`, `mars`, `jupiter`, `saturn`, `uranus`, `neptune` and `pluto`. Please see ‘[The APB themes](#)’ on page 38 for a discussion and examples.
- **apbBg:** Use the `apbBg` key to use one of the backgrounds provided by the APB. Recognized values are `mercury`, `venus`, `earth`, `mars`, `jupiter`, `saturn`, `uranus`, `neptune`, `pluto`, `ganymede`, `titan`, `phobos` and `europa`. The first nine, the names of the planets, are designs consistent with the corresponding themes; the other four, the moons, are additional gradient backgrounds. See ‘[Other APB Supplied Backgrounds](#)’ on page 39 for a discussion and examples.
- **use2D, use3D:** Selection of one of these two options determines the type of buttons that appear on the navigation panel. Specify `use2D` for a flat two-dimensional design, and `use3D` for graphic three-dimensional buttons. See the paragraph entitled ‘[Bottom Panel and Navibar: The Navigation Buttons](#)’ on page 41 for details.
- **userm:** The default font family is san-serif (`\sffamily`). Using this option switches back to roman (`\rmfamily`).
- **nosound:** Embedding and playing a sound associated with a layer is on by default. This option turns sound off. Useful for developing a talk when you don’t want to hear the sound all the time.
- **slideshow:** It is possible to automate the presentation: Put the document into full screen mode, click on the slideshow icon, and watch the points and pages change automatically. Use the command `\SlideshowTiming` to set the time between layers.
- **sideshow:** A sideshow is a sequence of toggle layers. A sideshow can be used to create a sequence of side events, such as a thermometer to track the progression through the presentation. The `sideshow` activates this feature.
- **paneloutline:** When this option is taken *and* either the `rightpanel` or `leftpanel` option is taken in the web package, a mini table of contents (or talk outline) is generated and appears in the panel.
- **annotslides:** When this option is taken, and a `paper` option is not taken in web, comments created by the `annot` and `authorannot` environments are written to a \LaTeX file, with an extension of `.ltx`. This file then used to create a PDF of thumbnails and comments on your document. See ‘[Create Thumbnails \(annots\)](#)’ on page 79 for details.

The option takes any one of four values `notes`, `nonotes`, `forauthor` and `forpublic`. With the first value, a file can be created with the author’s comments on each slide just under the thumbnail, a rectangular region is reserved for the participants to take notes on the talk; for the second value, a thumbnail file can be created with author’s comments on each slide under each thumbnail, no space is left for notes; the third value is meant to be used by the author during the talk, in the space provided, the author can list the main talking points of each page. `forpublic` is used for writing additional points to the audience (public) perhaps, beyond what is presented on the slides.
- **excludeannots:** The default behavior of the `annotslides` option is to insert the contents of the `annots` environment appears beneath the thumbnail of the slide as a caption description. A companion option to `annotslides`, this option removes the contents of the `annots` environment. See ‘[Create Thumbnails \(annots\)](#)’ on page 79 for details.

5.6. Font Sizes

APB uses the standard font sizes that come with the class being used, it is recommended that you use the `article` class. By using the package `extsizes` you can have access to 8pt, 9pt, 10pt, 11pt, 12pt, 14pt, 17pt, and 20pt. The `extsizes` package can be found on CTAN at `macros/latex/contrib/extsizes`.

5.7. Placement of Commands and Environments

For the APB, commands and environments can be placed in three locations: (1) the preamble; (2) between slides; (3) within a slide. Some commands and environments are required to appear in the preamble and nowhere else; some can appear in the preamble or between slides; others, only in a slide.

As each of the various commands and environments is introduced in this document, there is a statement about its proper location. Pay attention to these command locations, as placing a command or environment outside of its designated location may lead to a compile error, or else the functionality of the command may have no effect.

5.8. The `forscreen` and `forpaper` Environments

A presentation may not be only for the screen, but may be prepared as a paper handout. The web package has `forpaper` and `forcolorpaper` options for this purpose; choosing one of these options converts the document over to a paper document, formatted for the full length of the default paper size.

If there is going to be a paper version of your presentation, then you need to think about that aspect of your presentation as you develop your talk. The screen version of your talk may contain a variety of background graphics that you may not want to appear in your paper document. For the paper version, you don't want the navigation icons to appear, that makes no sense.

The two environments, `forscreen` and `forpaper`, (created using `comment.sty` by Victor Eijkhout) are used to place design and layout commands to be executed when the document is compiled for the screen, or executed when the document is compiled for paper.

```
\begin{forscreen}
....
....
....
\end{forscreen}
```

Environment Location: Anywhere.

Use the `forscreen` environment to insert commands or content meant only for the screen.

```
\begin{forpaper}
....
....
....
\end{forpaper}
```

Environment Location: Anywhere.

Use the `forpaper` environment to insert commands or content meant only for the paper document. For example, the `forscreen` environment may contain an animation, while the `forpaper` environment may contain a static image representing the animation.

Example 2: This example, appearing in the preamble, sets the background colors, background graphics, adds a logo watermark, and sets the default behavior of the viewer when it is in full screen viewing mode.

```
\begin{forscreen}
  \selectColors{textBgColor=cornsilk,panelBgColor=cornsilk}
  \template{mycoolgraphic.eps}
  \paneltemplate{mycoolgraphicforpanel.eps}
  \AddToTemplate{AEBLogo}
  \setDefaultFS{Trans=Random,timeDelay=5}
\end{forscreen}
```

6. The `\maketitle` and `\tableofcontents` Commands

As with any \LaTeX document, you can include `\maketitle` and `\tableofcontents` in the presentation. As with other APB content, each of these two commands must be enclosed in a `slide` environment, as was noted earlier in the discussion following [Example 1](#), page 12.

6.1. The `\maketitle` Command

The following is the proper syntax for including a title page.

```
\begin{document}

\begin{slide}
\maketitle
\end{slide}
...
\end{document}
```

The Navibar does not appear on the title page, by default. To force the appearance of the Navibar at the bottom of the page use the command `\NavibarOnFirstPage` in the preamble; this command is described in the paragraph on ‘[The Navigation Buttons](#)’, page 41.

When you take a panel option, but want to use the full width of the screen for the title page, use the `fullwidth` option of the `slide` environment, as the following illustrates,

```
\begin{slide}[fullwidth]
\maketitle
\end{slide}
```

The Navibar is *turned on* by default, in this case. If you don’t need the Navibar on, you need to turn it off manually:

```
\begin{slide}[fullwidth]
\NavibarOff
\maketitle
\end{slide}
```

Here, `\NavibarOff`, a web command, needs to be placed inside the environment, because the startup of the `slide` environment turns the Navibar off.

- **`\DeclareDocInfo`**

The preamble of your document should contain a number of keys that identify the document, including the title and author of the presentation. Some of this information is used to construct the title page, some is placed in the PDF, to be displayed in the ‘Description’ tab of the **Document Properties** dialog, which is accessed through the `Ctrl+D` accelerator key, or through the menu system `File > Document Properties`.

Information is passed through the `\DeclareDocInfo` command which takes a number of key-value pairs. This is a simple `xkeyval` interface to many of the [text macros](#) that are defined in `Web`.

```
\DeclareDocInfo{<KV-pairs>}
```

Command Location: Place in the preamble.

Section 6: The `\maketitle` and `\tableofcontents` Commands

Key-Value Pairs: The following is a description of the key value pairs.

1. `title`: The title of your presentation.
2. `author`: The author or authors of the presentation.
3. `subject`: The subject of the presentation. Optional, this appears only in the ‘Description’ tab of the **Document Properties**.
4. `keywords`: A list of keywords that describe your presentation. Optional, this appears only in the ‘Description’ tab of the **Document Properties**. Some search engines use this field.
5. `university`: The university or company the author represents.
6. `email`: The email address of the author. This appears on the title page, and becomes an email link.
7. `talkdate`: Date of the presentation.
8. `talksite`: Site of the presentation.
9. `copyrightyears`: Year(s) of the copyright of this publication, defaults to this year.
10. `prepared`: The date of preparation of the document, defaults to the day the file was compiled (\LaTeX ed).
11. `copyrightStatus`: If the `aebxmp` package is loaded for advanced metadata, this key allows you to set the copyright status. Possible values `True`, `False`, or blank (no value, or the key not listed at all) corresponding to `Copyrighted`, `Public Domain` and `Unknown`, respectively. The values of this key are case sensitive, so you must enter `True` and `False`, with the first letter capitalized. If `aebxmp` is not loaded, specifying this key does nothing.
12. `copyrightNotice`: If the `aebxmp` package is loaded for advanced metadata, this key allows you to set copyright notice, short text describing the copyright, perhaps,

```
copyrightNotice={Copyright D. P. Story, 2007}
```

If `aebxmp` is not loaded, specifying this key does nothing.

13. `copyrightInfoURL`: If the `aebxmp` package is loaded for advanced metadata, this key allows you to set the copyright info url, a url to a page on the web containing a more detailed description of the copyright. If `aebxmp` is not loaded, specifying this key does nothing.

The `talkdate` and `prepared` keys have labels. These two appear on the bottom row of the title page, as ‘Prepared: $\langle prepared-value \rangle$ ’ and as ‘Presented: $\langle talkdate-value \rangle$ ’. The two labels ‘Prepared:’ and ‘Presented:’ controlled by the following two commands.

```
\preparedLabel{\text_label}  
\talkdateLabel{\text_label}
```

Command Location: Both commands should be placed in the preamble,

Example 3: Example of usage of `\DeclareDocInfo`.

```

\DeclareDocInfo
{
  title=My First Presentation,
  author=D. P. Speaker,
  university=My University,
  email=dpspeaker@myu.edu,
  talkdate={Dec.\ 17, \the\year},
  talksite=The Talking University,
  subject=A basic APB template
  keywords={LaTeX, PDF, APB}
}

```

- Two tricks of importance: When the value contains a comma, then the whole value should be delimited by matching braces, as in the `talkdate` key-value in the **Example 3** above. The `\texorpdfstring` command is handy for giving alternate wording, when some of the \TeX commands do not transfer to the PDF's **Document Properties**. For example, the title might have been “A Discussion of e^x ”; this title should appear in `\DeclareDocInfo` as follows:

```
title=A Discussion of \texorpdfstring{\$e^x\$}{exp(x)},
```

Now the phrase, “A Discussion of $\exp(x)$ ” will appear in the ‘Title’ field of the **Document Properties**.

- There are command versions of the `\DeclareDocInfo` keys if you prefer to use them. They are `\title`, `\author`, `\subject`, `\keywords`, `\university`, `\email`, `\talkdate`, `\talksite`, `\copyrightyears` and `\prepared`. Each takes a single argument, for example, the following is the functional equivalent to **Example 3**.

Example 4: Command usage:

```

\title{My First Presentation}
\author={D. P. Speaker}
\university{My University}
\email{dpspeaker@myu.edu}
\talkdate{Dec.\ 17, \the\year}
\talksite{The Talking University}
\subject{A basic APB template}
\keywords{LaTeX, PDF, APB}

```

• The Title Page Structure

The title page is divided into three parts: top, middle and bottom.

- Title Top:** The content of the top is determined by the `\topTitlePage` command. (This command can be redefined, but it is not recommended.) The `\topTitlePage` command expands to three elements: the university (affiliation), the title, and the author(s), in that order vertically. These are the values of the keys `university`, `title` and `author` that appear in the `\DeclareDocInfo` command.
- Title Middle:** The `\optionalPageMatter` command is used to enter content into this part of the title page. This middle part is optional; if the `\optionalPageMatter` command does not appear in the preamble, then this part of the title page is empty.
- Title Bottom:** Bottom of the title page is controlled by the command `\titlepageTrailer` and consists of various document information entered in the `\DeclareDocInfo` command. By default, listed in `\titlepageTrailer` are the values of the `\DeclareDocInfo` keys `email`, `talkdate`, `talksite`, `copyrightyears`, as described above. The font size of this bottom part is set by the `\trailerFontSize` command, the default is `\footnotesize`. This command must be re-defined in the usual way: `\renewcommand{trailerFontSize}{\scriptsize}`, for example.

Section 6: The `\maketitle` and `\tableofcontents` Commands

Figure 2 shows the basic composition of the title page of an APB document. The title page elements are described as they relate to the key-values of `\DeclareDocInfo`, described on page 18.

- At the very top is the value of the `university` key. The color of this element can be set using the `universityColor` key of `\selectColors`, page 45.
- Next comes `title`, its color is controlled using the `titleColor` key of `\selectColors`.
- The author follows, which is the value of `author`. The color is set by the `authorColor` key of `\selectColors`.
- The APB logo is inserted using the `\optionalPageMatter` macro. Normally, this macro does nothing unless it is defined. In this example, we have

```
\optionalPageMatter
{%
  \begin{center}
    \begin{minipage}{.67\linewidth}
      \centering\includegraphics[scale=3.5]{APB_Logo}
    \end{minipage}
  \end{center}
}
```

- Finally comes the `\titlepageTrailer`, a macro that can be redefined (see `apb.sty` for its definition). This macro places the other elements at the bottom of the page:
 - The copyright year and email address, as given by `\email`. The color of the email address is set by `urlColor` of `\selectColors`. This color actually sets the color of all external URLs.
 - To the right, on the first line of the title page trailer is the value of `\talksite`. The color is the default color for text.
 - In the lower left is the date of the last revision of the document. The color is the default color for text.
 - In the lower right is the date the talk was given. The color is the default color for text.

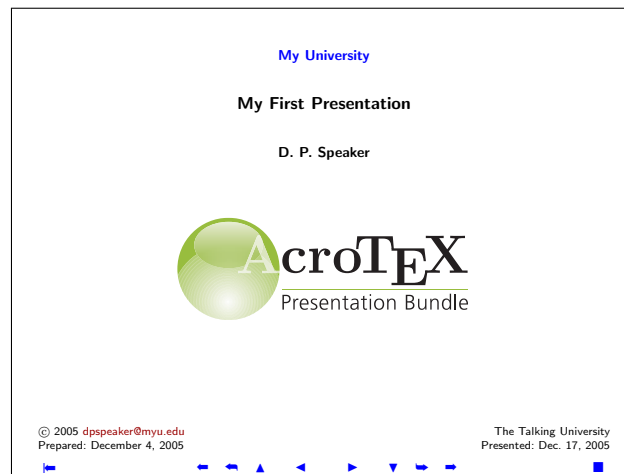


Figure 2: The Title Page

The title page layout is, of course, defined by the standard `\maketitle` command, which has been redefined in the APB package. The `\maketitle` has different behaviors depending on whether the document is being compiled for the screen or for paper.

- **Greater Control of the Top Title Page**

The three elements of the top title page are the values of the `university`, `title` and `author` keys that appear in the `\DeclareDocInfo` command. Corresponding to these, APB defines the commands `\universityLayout`, `\titleLayout`, and `\authorLayout` to format these three keys in a variety of ways.

ex17 A working example of the commands that follows can be found in `apb_ex17.tex`.

```
\universityLayout{<KV-pairs>}
\titleLayout{<KV-pairs>}
\authorLayout{<KV-pairs>}
```

Command Location: Place these (optional) commands in the preamble.

Key-Value Pairs: Each of these commands has a number of key-value pairs. The first thirteen are the same ones that appear in the description of `\sectionLayout`, ‘[Section Headings](#)’ on page 47. The rest are unique to these three commands. In the descriptions below, the word ‘element’ refers to the values of the keys `university`, `title` and `author`.

1. `fontfamily`: Font family to use with this element, permissible values are `rmfamily`, `sffamily`, `ttfamily`.
2. `fontseries`: Font series to use, possible values are `bfseries` and `mdseries`.
3. `fontshape`: Font shape to use: `upshape`, `itshape`, `scshape`, `slshape`.
4. `fontsize`: Font size to use with this element, permissible values are `tiny`, `scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `LARGE`, `huge`, `Huge`.
5. `halign`: Alignment of this element within its enclosing `\parbox`, permissible values are `l` (left aligned), `c` (centered), `r` (right aligned). See [Example 5](#) for a visualization of the effects of the `halign` key.
6. `color`: The color of the section title, this can be any named color. The default is blue for `title`, and black otherwise.
7. `special`: Through this key, you can specify predefined layout for the title elements. Permissible values: `shadow`, `framebox`, `colorbox`, `fcolorbox`, `frameboxfit`, `colorboxfit`, `fcolorboxfit`, `custom` and `default`.

Custom titles can be created by specifying a value of `custom`. In this case, APB uses the commands `\customUniversity`, `\customTitle` and `\customAuthor`. These are macros that take one argument, the code for designing the title. The title is referred to as #1. Depending on how these custom titles are defined, the other keys may not be obeyed. See an [Example 37](#).
8. `framecolor`: The color of the frame surrounding the subject when the `special` key has a value of `framebox`, `fcolorbox`, `frameboxfit` or `fcolorboxfit`.
9. `bgcolor`: The background (fill color) of the box enclosing this element, when `special` has a value of `colorbox`, `fcolorbox`, `colorboxfit` or `fcolorboxfit`.
10. `shadowcolor`: The color of the shadow, when `special` has a value of `shadow`.
11. `beforeskip`: The amount of skip before the title element.
12. `afterskip`: The amount of skip after the title element.
13. `usefont`: Through this key it is possible to specify an arbitrary font and font size. The key takes five parameters, for example, `usefont={OT1}{cmdh}{m}{n}{{16}{16pt}}`. The first four are the arguments of the \TeX 's `\usefont`, encoding, family, series and shape. The last argument are the arguments of the \TeX 's `\fontsize`, size and baselineskip.

Section 6: The `\maketitle` and `\tableofcontents` Commands

If the fifth parameter is empty, no font size is specified, the current default sizes are used, e.g., `usefont={OT1}{cmdh}{m}{n}`.

14. `hproportion`: Each of the elements (`university`, `title`, `author`) lie in their own `\parbox`, the width of this box is determined by the value of this key, as a proportion of the total `\linewidth`. The default for all three is `.7`. This value can be set to get more or less “natural” line breaks, without having to insert a new line with a `\\`. See [Example 5](#) for a visualization of the effects of the `hproportion` key.
15. `xhalign`: The `\parbox` of each of the three elements are also placed in a `\makebox`, additional control over positioning can be had by setting this key, which sets the positioning parameter of `\makebox`. The default value for `xhalign` is `c`, the element is centered. See [Example 5](#) for a visualization of the effects of the `xhalign` key.

There is one other title page parameter that effects the layout.

```
\topTitlePageProportion{<0..1>}
```

Command Location: Place this (optional) command in the preamble.

Parameter Description: The top part of the title page is enclosed in a big `\parbox` with depth set to a proportion of `\textheight`. This proportion is set through the command `\topTitlePageProportion`. The argument of this command should be a number between 0 and 1, obviously a value of 0 makes no sense. The default value is set by the web package to `.33`, i.e., `\topTitlePageProportion{.33}`.

ex17 **Example 5:** [Figure 3](#) gives a representation of the page layout of the title page. The big `\parbox` of depth equal to the proportion of `\textheight` determined by `\topTitlePageProportion` is shown as a blue box. Each of the three top title elements are enclosed in a `\makebox`, shown in yellow. Inside this `\makebox`, the top title elements are placed in a `\parbox`, shown in gray. The image shown in [Figure 3](#) came about as a result of the following commands in the preamble:

```
\topTitlePageProportion{.5}
\universityLayout{halign=l,color=red}
\titleLayout{halign=r,xhalign=r,hproportion=.4}
\authorLayout{color=webbrown}
```

In [Figure 3](#), notice that the `university` is left aligned within its `\parbox`. For `title`, the proportion is changed from the default of `.7` to `.4`, this is manifested by the shorter gray box (which represents the underlying `\parbox`); `halign` and `xhalign` are both set to `r`, so the title appears right aligned within its `\parbox`, and the `\parbox` is right aligned within its `\makebox`, understand? Finally, for the `author` key, we change only the color. Cool!

Example 6: This example illustrates the custom feature for creating title elements. This example places a gradient box around the title. It uses the `pstricks` and `pst-grad` packages. See the example file `apb_title.tex` for several examples of a custom title.

In the script below, we use the `\apb@frameboxHead` command for placing a frame around the title (used by the `framebox` value of the `special` key), then covering that content using the `\psframebox` from `pstricks`.

```
\makeatletter
\customTitle{\setlength{\fboxrule}{0pt}\psframebox[\linewidth=0pt,%
  fillstyle=gradient,gradangle=30,gradbegin=yellow,%
  gradend=red]{\apb@frameboxHead{#1}}}
\makeatother
\titleLayout{special=custom,hproportion=.5}
```

Section 6: The `\maketitle` and `\tableofcontents` Commands

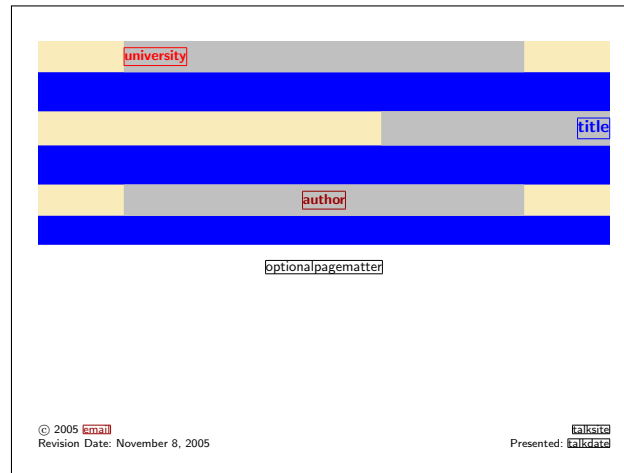


Figure 3: The Title Page Layout Structure

6.2. The `\tableofcontents` Command

The table of contents presented an interesting set of problems in the development of the [AcroTeX Presentation Bundle](#). There is a tight control over the creation of pages, content is not allowed to flow from one page to another; if it does, the \LaTeX compile stops, and the document author is asked to create a new slide for the overflow content.

In a presentation with many sections, subsections and subsubsections, it may be the case that the table of contents will be more than one page long. The table of contents is automatically generated, and the document author cannot manually create additional slides for the table of contents.

It was concluded after many hours of meditation, that the break up of the table of contents into several slides must be done programmatically. To aid in the solution to the problem, the `\splitTocAfter` command is used.

```
\splitTocAfter{<natural-number>}
```

Command Location: Use this command in the preamble.

Parameter Description: Use `\splitTocAfter` to set the number of lines to appear on each of the table of contents slides. The default is `\splitTocAfter{10}`, that is, break the table of contents slide into another slide after each 10 lines written.

Example 7: Here, we change the setting of `\splitTocAfter` from its default to 8 lines per table of contents slide.


```

\splitTocAfter{8}
\begin{document}

\begin{slide}
\maketitle
\end{slide}

\begin{slide}
\tableofcontents
\end{slide}

\begin{slide}
Slip sliding away...
\end{slide}
\end{document}

```

By default, the outline of the talk or presentation is in the form of “talking points”. As the ‘Next’ button is pressed, each section, subsection and subsubsection appears (for discussion). If this behavior is not desired for your document, include the `\turnOcOff` command within the slide; the effects will be local to the slide.

```

\begin{slide}[narrower, fontsize=Large]
\turnOcOff
\tableofcontents
\end{slide}

```

The APB lists the items in the table of contents through the third level using the structured list commands `\dPt`, `\ddPt` and `\dddPt`, as discussed in ‘[Structured Lists](#)’. These commands use a ding to indicate its level. The dings for the table of contents and the other dings for the structured list can be set through the `\selectDings`, discussed in the paragraphs on ‘[Structured Lists](#)’, on page 28.

The default definition is that the dings for the table of contents are the same as the dings for `\dPt`, `\ddPt` and `\dddPt`; however, if a different set of dings is desired, then use `\selectDings` in the preamble, like so

```

\selectDings
{
    dDingToc=\ding{082},
    ddDingToc=\ding{079},
    dddDingToc=\ding{254}
}

```

Set the color of these dings through `\selectColors`, see ‘[Selecting Colors](#)’ on page 45.

There are command versions of the keys `dDingToc`, `ddDingToc`, and `dddDingToc`, they are

- `\dDingToc{<ding>}`
- `\ddDingToc{<ding>}`
- `\dddDingToc{<ding>}`

7. Inserting Content with the `slide` Environment

At the heart of any presentation is the *content*: what it is to be said, and how it is to be presented. All content must be placed on a *slide*. The term “slide” is an archaic term left over from the days when the content of a presentation was placed on a transparency (i.e., a slide) and projected onto a screen.

Section 7: Inserting Content with the `slide` Environment

For an electronic presentation, *using traditional methods*, a slide may consist of many physical pages. As the presenter pages through the document, talking points may be added to the talking points of the previous page. This continues until the slide is completed, at which point a new slide is begun. An illusion is created in the mind of the viewer that the talking points appear on the page one after the other, but this is not so, the presenter is merely paging through the document. As a result, the document may have a large number of pages, even though it has very few slides.

For APB, a slide is the same as a physical page; there is a one-to-one correspondence between the two.

7.1. The `slide` Environment

```
\begin{slide}[\langle KV-values \rangle]
....
slide content
....
\end{slide}
```

Command Location: In body of the document.

Key-Value Pairs: The argument takes key-value pairs to design the slide. Except for `fullwidth` and `sectitle`, these are the same keys as specified in the `\setDefaultSlides` command. Setting these keys will *locally* override APB defaults, which are set by `\setDefaultSlides`; they will be in effect for this slide only.

1. `fullwidth`: A Boolean, which if true, commands APB to remove the navigation panel and to replace it with a fullwidth text screen. Useful for presenting wide pictures or equations on the screen. The slide system resets itself for the next slide.
2. `sectitle`: When the `fullwidth` option is chosen and a new section begins within the slide, use `sectitle` the title of the section begun within the slide. This is for the purpose of getting the running header to properly reflect the current section.
3. `narrower`: A Boolean, which if true, narrows the margins of the slide, perhaps for a more eye-pleasing presentation.
4. `image`: A Boolean, which if true, the background of this slide should be a fullwidth image inserted by the `\insertImage` command. (See ‘[Full Page Images](#)’ on page 30.) This is a workaround for a bug that I haven’t been able to get a good solution to. When this option is not taken in a fullwidth slide (assuming there is a panel) and `\insertImage` is used to insert a picture background, two slides with the same background image are created.
5. `indent`: A length that is used by `narrower` to increase the left and right margins. The default value is `\apbIndent`, which on startup is set to 20pt.
6. `vcenter`: A Boolean, which if true, attempts to center the content of the slide vertically. The normal behavior is not to center.
7. `fontsize`: The default size of the font to appear within this slide. Permissible values are `tiny`, `scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `LARGE`, `huge` and `Huge`. These correspond to the standard font sizes. The default is the class default size.
8. `fontseries`: Sets the default series for this slide. Permissible values are `bfseries` and `mdseries`. The default is the class default font series.
9. `color`: Sets the color of text for this slide. Any name of any named color is permissible. The default is T_EX’s default color, usually black.

Section 7: Inserting Content with the `slide` Environment

- Because `slide` is an environment, any definitions within the slide are local to that slide. Global changes should take place between slides, or in the preamble.

Example 8: This example sets some slide options and creates a page of content.

```
\begin{slide}[fontseries=bfseries,fontsize=Large,narrower]
\section{My Great Talk}

Hello World This talk is brought to you by the AcroTeX
Presentation Bundle.

\end{slide}
```

Example 9: Here we assume a panel option is taken, but we want to use the entire width of the slide for the content.

```
\begin{slide}[fullwidth,sectitle=My Wide Talk,fontseries=bfseries,
              fontsize=Large,narrower,indent=30pt,color=red]
\section{My Wide Talk}

Hello World This talk is brought to you by the AcroTeX
Presentation Bundle.

\end{slide}
```

7.2. Simple Talking Points and Structured Lists

In this section, we discuss ways you can present your talking points.

- **Simple Talking Points: `\Bld`**

This easiest way of building “a talking point” in a presentation is via the `\Bld` command.

ex2 **Example 10:** The following illustrates the usage of `\Bld`, the text would appear within a `slide` environment. See the tutorial file `apb_ex2.tex` for a working example.

```
\Bld Welcome to the wonderful world of \Bld\textcolor{red}{AcroTeX}.

The APB can be used to create what? \Bld\textbf{A Quality Presentation!!}
```

This example creates three phrases that appear one after the other in the order they were created. The three phrases are: (1) “Welcome to the wonderful world of”; (2) “**AcroTeX**.”; and (3) “**A Quality Presentation!!**” As you click on the ‘Next’ button, these phrases will appear in the order (1), (2) and (3).

- **Important:** The scope of `\Bld` is delimited by the *next* `\Bld` command.

```
\ocOff
```

Use `\ocOff` to limit the scope of the last `\Bld` command.

Example 11: This example illustrates how to terminate the scope of `\Bld`

```
\Bld Welcome to the wonderful world of \ocOff\textcolor{red}{AcroTeX}.

The APB can be used to create what? \Bld\textbf{A Quality Presentation!!}
```

Section 7: Inserting Content with the slide Environment

Here there are only two talking points: (1) “Welcome to the wonderful world of” and (2) “**A Quality Presentation!**”. The other text “**AcroTeX**.\par The APB can be used to create what?” is visible on the page. The talking points appear as the ‘Next’ button is pressed.

☛ At the end of each slide, an `\ocOff` command is automatically executed to terminate the last `\Bld`.

• Structured Lists

Talks are often presented as a series of bullet or talking points. For this purpose, APB provides `\dBld`, `\dPt`, `\ddPt`, `\Item`, `\IItem` and `\IIItem`. There are also environment forms for each of these as well.

These commands can be made to dim (down), dim up or not to dim at all depending on the options or commands that have been issued. To review, to dim up use the `dim` option for the whole document or `\DimOn` to turn on dimming locally; if you want a dimming up effect, use the `dimup` for the whole document, or `\DimUp` to turn on the dim up effect locally. See ‘Options of APB’ on page 15, for more details.

```
\dBld[⟨KV-pairs⟩] ⟨text⟩ \par
\dPt[⟨KV-pairs⟩] ⟨text⟩ \par
\Item[⟨KV-pairs⟩] ⟨text⟩ \par
\ddPt[⟨KV-pairs⟩] ⟨text⟩ \par
\IItem[⟨KV-pairs⟩] ⟨text⟩ \par
\dddPt[⟨KV-pairs⟩] ⟨text⟩ \par
\IIItem[⟨KV-pairs⟩] ⟨text⟩ \par
```

Command Location: These commands must appear within a `slide` environment

Key-Value Pairs: The argument takes key-value pairs to design the item.

1. `textcolor`: The defined color. The color of the text to appear in this item. Defaults are set depending on the level.

Set default color through `\selectColors`, see ‘Selecting Colors’ on page 45.

2. `dingcolor`: Any defined color. The color of the ding. Defaults are set depending on the level.

Set default color through `\selectColors`, see ‘Selecting Colors’ on page 45.

3. `print`: (APB 2.1) A choice key, the permissible values of `print` are `true`, `false`, `always`, `never`, and `whenever`. The first two, `true` and `false`, are legacy values from APB 2.0, see below; the third and fourth, `always` and `never`, are equivalent to `true` and `false`, respectively. The fifth choice, `whenever`, is new, in this case, the layer prints only if it is visible. The terms `always`, `never`, and `whenever` roughly correspond to the terms used in the user interface.

`print`: (APB 2.0) A Boolean, which if `true`, the default value, this layer will be printed even if it is hidden. Set `print=false` if you do not want this layer to be printed while hidden. It will be printed while visible.

4. These commands obey the key-value pairs of sub-page navigation, as listed in Table 1, page 10. When using a dim up effect, the transition effects (`nexttrans` and `prevtrans`) only work for the material that is not dimmed.

ex3 Example 12: Figure 4 is taken from the working example, `apb_ex3.tex`, meant to illustrate the basic functionality of the APB lists. The figure to the right depicts the results of this listing.

☛ The items created by these commands *must be separated* by a `\par` (or a blank line), as in Example 12.

The talking point commands `\dPt`, `\ddPt` and `\dddPt` each have a ding that appears at the beginning of the point. These dings are defined by the `\selectDings` command, discussed ‘Selecting Dings’ on page 44.

Section 7: Inserting Content with the slide Environment

```
\begin{slide}[fontsize=LARGE,fontseries=bfseries]
\section{My Talking Points}
\dBld This is the zeroth level
\DPt This is the first level
\Item Continuation of first level
\ddPt This is the second level
\IIItem Continuation of second level
\dddPt This is the third level
\IIIItem Continuation of third level
\end{slide}
```

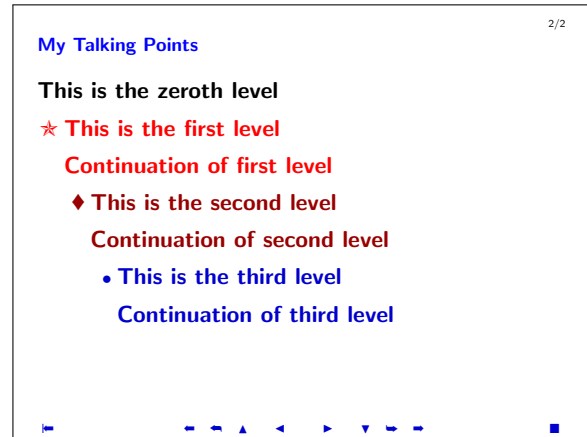


Figure 4: Listing for Example 12

When dimming is in effect, items are dimmed up or down using a mixture of the text color (as set by the `textcolor` key) and the background color of the page. The default is to take a mixture of 25% of the text color and 75% of the background color to form the dimmed color. This value can be changed by `\dimlevel`.

```
\dimlevel{<number>}
```

Command Location: When placed inside a slide environment, the changes are local to that slide; otherwise, the changes are global.

Parameter Description: The `<number>` should be an integer between 0 and 100. The number is interpreted as a percentage: Let $p = \langle number \rangle$, then $p\%$ of the text color is mixed with $(100 - p)\%$ of the background color to form the dim color. The default is 25.

There is a technical reason for this. To create the dimming effect, two layers need to be created for the text. One layer is for the text (highlighted in the default color) the other layer is for the dimmed text. The text is read in by these commands as its argument, delimited by `\par`.

- Because the text is actually an argument of the command, you cannot use any verbatim commands such as `\verb` in the text. There are environment versions of each of these commands that can be used as a work around.

Another special effect that is available through these dimming list commands is the `\step` command.

```
\step{<text>}
```

This command is used to “step” through a dimming point (`\DPt`, and so on). If `\step` is used, then the union of all arguments must be the entire body of the dimming point. An example follows to illustrate.

ex3 Example 13: Step through a dimming point.

```
\dPt \step{Let's make our first point.} \step{Now make our second.}
\step{One list important point.}
```

The dimming is a bit tricky with layers, and has its limitations. One limitation is working with verbatim text. As mentioned earlier, you cannot use the `\verb!...!` command with these structured talking point commands. It is for this reason the environment versions of the dimming talking points were defined.

- For the environment versions, a `\par` does not delimit the end of the text (as is the case for the command forms); the end of the environment does that.

ex3

Example 14: Here is a snippet taken from `apb_ex3.tex` and illustrates the environment versions of the dimming commands, and how to handle verbatim. See this tutorial file for additional examples and tricks.

```
\begin{dPt}[textcolor=webgreen]
Let's try a verbatim like \verb!$%! .
\end{dPt}
\begin{dPt}
\step{Can we step through a verbatim?} \step{Well, let's see:}
\step{\verb!3 # $ }!} \step{and \verb!e^{x^2}\sin(x)! .}
\end{dPt}
\begin{ddPt}
Or, we can try a display verbatim, see if that works:
\begin{verbatim}
    x^2 + y^2 = 1
\end{verbatim}
\end{ddPt}
```

7.3. Full Page Images

Images can be imported using `\includegraphics` in the usual \LaTeX way. The images cover a rectangular subregion of the text region of the slide. There may, however, be an occasion for which you wish to have an image cover the entire page, with a caption. This section describes the necessary commands for creating full page images. History or Art History classroom presentations, for example, may need the full page image to create a larger image, making it easier to see from the audience. The full page images can be used to create, in essence, a *photo album*; taken in conjunction with `annots` feature of APB, there are a number of possibilities available to the creative and energetic mind.

The commands relevant to full page images are `\insertImage` and `\captionLayout`. The former inserts the full page image on the slide, with caption, and the latter is used to set the format for the caption.

```
\insertImage[<KV-pairs>]{<graphics_path>}
```

Command Location: Use this command only within a `slide` environment. Use this command only once per slide.

Key-Value Pairs: The `\insertImage` command has a number of different key-value pairs.

1. `fit`: Permissible values are `width` and `height`. Use `width` to fit the image to the width of the page, and `height` to fit the image to the height of the page.
2. `caption`: The value of this key takes a string that is to be the caption for the image.
3. `vshift`: Vertically shifts the image, `vshift=-20pt` shifts the image 20 points down.
4. `hshift`: Horizontally shifts the image, `hshift=20pt` shifts the image 20 points to the right.
5. `inclgraphics`: Use this key to pass any optional parameters to `\includegraphics`, the underlying graphics command from the `graphicx` package. Multiple arguments are separated by commas and enclosed in braces; an example of multiple arguments is `inclgraphics={hiresbb,clip}`.

The `fit` key is a convenience to set two (`fit` and `width`) of the optional parameters of `\includegraphics`. When `fit=width`, the `width` key of `\includegraphics` is set to `\paperwidth`, and `fit=height` sets the `height` key of `\includegraphics` to a value of `\paperheight`.

`\captionLayout{<KV-pairs>}`

Command Location: When used in the preamble or between slides, the changes are global, when used within a `slide` environment, the changes are local to that slide.


Key-Value Pairs: The `\captionLayout` command has a number of different key-value pairs.

1. `fontfamily`: Font family to use with this caption, permissible values are `rmfamily`, `sffamily`, `ttfamily`.
2. `fontseries`: Font series to use, possible values are `bfseries` and `mdseries`.
3. `fontshape`: Font shape to use: `upshape`, `itshape`, `scshape`, `slshape`.
4. `fontsize`: Font size to use with this caption, permissible values are `tiny`, `scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `LARGE`, `huge`, `Huge`.
5. `halign`: Alignment of the caption within its enclosing `\parbox`, permissible values are `l` (left aligned), `c` (centered), `r` (right aligned). See [Example 5](#) for a visualization of the effects of the `halign` key.
6. `hproportion`: The caption lies in its own `\parbox`, the width of this box is determined by the value of this key, as a proportion of the total `\linewidth`. The default for all three is `1`. This value can be set to get more or less “natural” line breaks, without having to insert a new line with a `\\`.
7. `xhalign`: The `\parbox` of the caption is also placed in a `\makebox`, additional control over positioning can be had by setting this key, which sets the positioning parameter of `\makebox`. The default value for `xhalign` is `c`, the `\parbox` is centered within the `\makebox`.
8. `color`: The color of the text of the caption. This can be any named color. The default is black.
9. `special`: Through this key, you can specify one of the predefined layouts for the caption. Permissible values are `shadow`, `framebox`, `colorbox`, `fcolorbox`, `frameboxfit`, `colorboxfit`, `fcolorboxfit`, `custom`, `default`.

Custom captions layouts can be used by specifying a value of `custom`. In this case, APB uses the command `\customCaption`. This macro takes one argument, the code for designing the caption. The caption string is referred to as `#1`. Depending on how these custom caption layouts are defined, the other keys may not be obeyed. See [an Example 37](#).


10. `framecolor`: The color of the frame surrounding the caption when the `special` key has a value of `framebox`, `fcolorbox`, `frameboxfit` or `fcolorboxfit`.
11. `bgcolor`: The background (fill color) of the box enclosing this caption, when `special` has a value of `colorbox`, `fcolorbox`, `colorboxfit` or `fcolorboxfit`.
12. `shadowcolor`: The color of the shadow, when `special` has a value of `shadow`.
13. `beforeskip`: The amount of skip before the caption.
14. `afterskip`: The amount of skip after the caption.
15. `usefont`: Through this key it is possible to specify an arbitrary font and font size. The key takes five parameters, for example, `usefont={OT1}{cmdh}{m}{n}{16}{16pt}`. The first four are the arguments of the \LaTeX 's `\usefont`, encoding, family, series and shape. The last argument are the arguments of the \LaTeX 's `\fontsize`, size and baselineskip.

If the fifth parameter is empty, no font size is specified, the current default sizes are used, e.g., `usefont={OT1}{cmdh}{m}{n}`.

 **Example 15:** The demos for this feature are `apb_pictures.tex` and `apb_pictures1.tex`, found in the `examples` folder. The file `apb_pictures1.tex` shows you how to write to the navigation panel using `\atmiddlepanel`, this is useful for writing comments on the content of the image.

A full-page image (and its caption as well) is inserted as an overlay (or template) using the `\template` and `\fullwidthtemplate` commands of `Web.APB` actually redefines some of the internal commands of `web` to acquire greater control over the placement of the image. `\insertImage` must therefore be placed within a `slide` environment to make these redefinitions local to the group containing the contents of the slide; background graphics and full-page images can be used in the same document, as a consequence of this placement convention.


Full-page images can be used with the panel options. With panels you can also use the `fullwidth` option of the slide environment. This gives the ability to create a presentation with talking points, and full-page images (and full text screen images as well).

 For a document that uses a panel option, when a full-page image is used with a `fullwidth` slide, use the `image` option as well.

```
\begin{slide}[fullwidth,image]
\insertImage[fit=height,caption=A Horse]{horse}
\captionLayout{%
  color=white,bgcolor=black,beforeskip=.25in,hproportion=.9,
  halign=l,special=colorboxfit
}
\end{slide}
```

Because the full-page images are overlays, you can write on top of the images with text, with talking points, with other graphics inserted by the usual `\includegraphics` command. Some effort needs to be made to avoid writing on top of a critical portion of the image, however; a good `dvi` previewer is helpful in this regard. The demo file `apb_pictures.tex` uses the package `textpos`, by Norman Gray, to place text and a framed box with text. You may find this package useful for this purpose, but it is certainly not a required package for this feature.


- **Moving the Panel**

 For a document that is compiled with a panel option, it is possible to move the panel from one side to the other, as is demonstrated in `apb_pictures1.tex`. To make this transition easier, there are a few useful commands:

```
\paneltoleft
\paneltoright
\resetpanel
```

Command Location: Place these commands between slides.

Command Description: `\paneltoleft` moves the panel to the left side, `\paneltoright` moves the panel to the right side and `\resetpanel` moves the panel to the side specified in the original panel option, on the right for `rightpanel` and on the left for `leftpanel`.

 When you move the panel from its default side, the margins on the text screen are not correct, so typesetting in the text screen may not look correct. The feature of moving the panel is only useful if you have an image in the text screen, and write to the navigation panel. Additionally, the caption feature works correctly, and using `textpos` works as well.

As mentioned at the beginning of the section, `APB` has an annotation/thumbnaill feature that can be used to create thumbnails of each slide along with the annotations. This feature may be useful for creating notebooks of photos with memories, a photo album.

- **Full-Page Image and Printing**

The full-page image is incompatible with the `forpaper` and `forcolorpaper` options of Web. With these options, a design decision was made to remove all overlays for the printed page, and as a result, the full-page images are removed. Should you want a printable version of your presentation containing full-page images, use the `annotation/thumbnail` option instead to create a printable handout, this may be a superior choice anyway.

8. Navigating a Presentation

APB has two ways of navigating through a presentation: (1) by using an elaborate set of buttons; (2) by using left/right arrow buttons or left/right clicking of the mouse. The first was the one method available in the original version of APB, while APB introduces the second method. Which method should you use? This section addresses that question.

8.1. Sub-Page Navigation

When giving a presentation in front of an audience, use the `apbnavi` option. This will enable you to move forward through your talk with minimal effort. If you have a wireless presenting device, you can use it to move through the slides and the different layers of the slides remotely. *Sub-page navigation only becomes operational in fullscreen mode.*

If the `apbnavi` option is used, there is no real need for a left or right panel. Without the panel, there is no need for the large number of buttons available to you in the navigation bar (when the `navibar` option is taken in the web package). Should you choose to use the panel, there is again no need for the elaborate button set built into the panel. Therefore, when in fullscreen mode, and if you choose the `spnavibar` option for APB, there is a limited button set available for navigation, buttons to go to the next page, the previous page, the first page, and to exit fullscreen.

The navigation bar at the bottom of the screen and the button set in the panel are now in their separate layers. When not in fullscreen, you see the full set of buttons, when in fullscreen you see the reduced set of buttons (both for screen and panel). The presence or absence of these buttons is controlled by two navigation options.

There are two options that produce navigation buttons: `navibar` for web and `spnavibar` for APB. Here's how they work. If you take `navibar` option, only the navigation buttons (for the screen) appear, then you go into fullscreen mode, assuming the `spnavi` option is in effect, there is no navigation bar. When the `spnavibar` option of APB is taken, you get the navigation bar for web, visible when not in fullscreen mode, and the navigation bar (with reduced number of buttons) when in fullscreen mode. If neither of the navigation options is taken, no navigation bars are displayed. (In this case, you're on your own!)

- **`\ClearPagesOnClose` and `\DoNotClearPagesOnClose`**

When you navigate backward in fullscreen mode to a previous page, the current node becomes the one at the top of that page. If you've already exposed your talking points for that page, navigating forward from that point does not seem to do anything; you are, in fact, just making layers already visible, visible. It is not comfortable to navigate in this situation. APB, therefore, modifies this natural behavior of sub-page navigation.

The default behavior of an APB document using sub-page navigation is to reset all layers on the page to their initial state when the page closes. Consequently, when you navigate to a previous page, all layers are reset, and navigating forward from that point shows them again. The resetting of layers is turned on by default, but can be turned off and on again using the commands that follow.

```
\ClearPagesOnClose
\DoNotClearPagesOnClose
```

Command Location: Use in preamble or between slides, changes are global. Within a slide, changes are local to that slide.

Command Description: `\ClearPagesOnClose` turns on the default behavior of resetting all layers to their initial state on page close, while `\DoNotClearPagesOnClose` turns off this behavior.

- **Sub-Page Navigation in a Browser**

Version 8 of **Acrobat** and **Adobe Reader** have an undocumented feature, which I call “pseudo-fullscreen mode,” for documents in a web browser in fullscreen mode. When a document opens in fullscreen (or is put into fullscreen using `Ctrl+L`, for example) the hand tool has a little up-arrow on it. This is the icon for fullscreen navigation.⁵

While the mouse cursor shows fullscreen navigation control in the browser, left-clicking on the mouse advances through the page or layer. This new control in the browser makes it possible to produce an Internet-based presentation.⁶

There are some differences between fullscreen mode and “pseudo-fullscreen mode” in a browser. The browser version of fullscreen does not obey page transitions, but does obey the sub-page navigation transitions. I have discovered that Version 8 does not obey the **Dur** key, meaning that layers cannot be timed to change as they can be outside the browser. These points should be kept in mind when designing a browser-based presentation.

8.2. Navigation via Buttons

There are two entirely different mechanisms for navigating: By buttons (in this case, the APB keeps track of the current location on each page, and it is JavaScript that turns on the layers or off the layers), or by sub-page navigation (here the process is more automated, as you move from node to node).

The elaborate button set is essential for navigating through an APB document while not in fullscreen mode. The document author may use the buttons to move around in the document while in development phase, or during a talk, may find it necessary to drop out of fullscreen mode, in which case the buttons are there to aid navigation.

If the presentation is not before an audience, but is distributed over the Internet, for example, a decision must be made: To use sub-page navigation (which is only in effect in fullscreen mode) or not. If not, then the full button set is essential for the document consumer to navigate through the document.

APB provides you with the option of using sub-page navigation, with or without buttons, or not to use it, with a full range of buttons. This is a design choice of the document author.

9. Customizing APB

When you are first designing your presentation, you must make a series of decisions apart from the content of your presentation. In this section we describe these decisions, which form a customization of your APB document. The list below is more-or-less in a “natural” order.

- ▶ The decisions you make concerning the design and layout of your presentation are not “engraved in stone”: they can be changed at any time.

Design Decisions:

1. Choosing the Screen Dimensions: [Section 9.1](#)
2. Choosing between Fullwidth and Paneled Layouts: [Section 9.2](#)

9.1. Choosing the Screen Dimensions

The first thing to be done is to decide what screen dimensions you want for your presentation. These dimensions must be chosen for easy viewing on a projected screen, or on a computer monitor, and, perhaps, wide enough to fit your widest equation or figure that you might want to present. Some experimentation might be needed to finally settle on the desired dimensions.

The web package provides several options and commands for this purpose. Choosing any of the web options `designi-designvi` gives preset dimensions which are satisfactory for screen viewing.

- ▶ For a screen resolution of 1024×768 pixels, I prefer `designv`.

⁵For versions previous to version 8, the little up-arrow does not appear in the same situation, fullscreen in a browser.

⁶Remember, Version 8.0 or later.

Section 9: Customizing APB

You can create custom dimensions as well by using `\margins` and `\screensize`. This syntax for these are

```
\margins{<left>}{<right>}{<top>}{<bottom>}
\screensize{<height>}{<width>}
```

Example 16: The following are the definitions for `designv`.

```
\margins{.25in}{.25in}{24pt}{.25in}
\screensize{4.5in}{6in}
```

9.2. Choosing between Full Width and Paneled Layouts

Fundamentally, the document author must choose between having a presentation that utilizes the full width of the screen page, or to have a navigation panel located either on the left or the right.

Figure 5 shows the full width layout and a paneled layout. In the first case, navigation control appears at the bottom of each page, and in the second case, the control buttons appear in the (navigation) panel. ‘[Bottom Panel and Navibar: The Navigation Buttons](#)’ on page 41 discusses the customization of the navigation control.

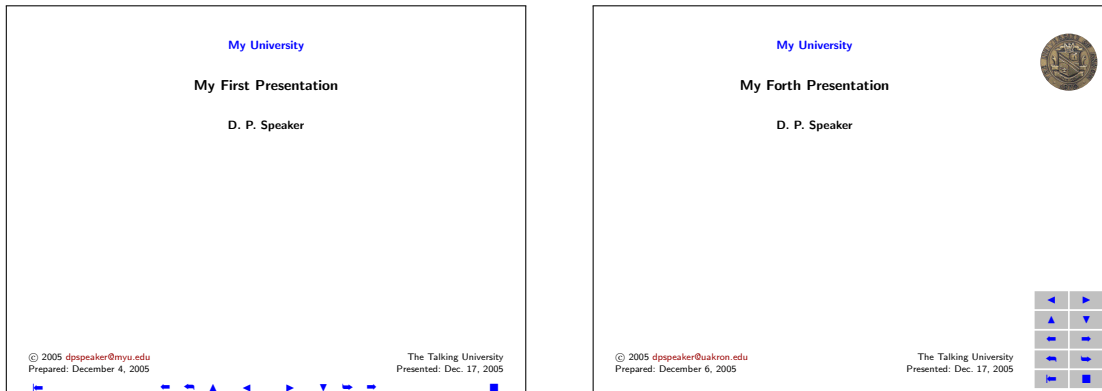


Figure 5: Fullwidth Layout versus Paneled Layout

- **Full Width Layout**

The full width layout is the default, be sure to include the `navibar` option to obtain navigation control over the presentation.

Example 17: Full width Layout: This example sets up a rather generic full width layout. Notice the inclusion of `navibar`, to get the navigation control bar, and `usetemplates`. This last option is optional, but it provides support for creating background colors and graphical backgrounds.

```
\usepackage[
  dvips,
  designv,
  navibar,usetemplates
]{web}
\usepackage{apb}
```

- **Paneled Layout**

The paneled options give a much more elegant looking presentation than does the full width layout. To obtain the panel, use either the `rightpanel` or `leftpanel` option of the web package.

Example 18: Paneled Layout: This snippet shows a standard setup for getting the paneled layout. We use the `rightpanel` option of `web` in this case.

```
\usepackage[
  dvips,
  designv,
  navbar,rightpanel
]{web}
\usepackage{apb}
```

Full width is nice because it maximizes the screen width that can contain the content of the presentation. Should a panel option be taken, all is not lost.

Suppose the document author chooses a panel option, and has a slide that needs to contain a lot of content. APB has a provision for easily removing the panel and put the slide into “full width mode”.

Example 19: Using the `fullwidth` option of `slide`: The premise in this example is that the document author has chosen a panel option for `Web`, but wants to use the full screen width. In this case, the `fullwidth` option of the `slide` environment is used:

```
\begin{slide}[fullwidth]
Consider the following picture:
\begin{center}
\includegraphics{mywidepic.eps}
\end{center}
\end{slide}
```

ex4 A working example of the `fullwidth` slide option can be found in `apb_ex4.tex`.

9.3. Choosing Background Color and Graphics

So far, the examples presented here have been with a plain white background. You can jazz up your presentation by adding a background color, a background graphic, or a combination of the two.

The APB comes with nine themes, these are color coordinated background graphics, as well as several other “generic” background graphics for you to choose from. You can, of course, use your own graphic.

- **General Concepts and Methods**

In this paragraph general methods of defining background colors and graphics. We begin with colors, then cover graphics.

Most all colors are be defined through the `\selectColors` interface, see the section titled ‘[Selecting Colors](#)’ on page 45. The four keys `textBgColor`, `fullwidthBgColor`, `panelBgColor` and `slideColor` are of interest here.

ex4 **Example 20:** Set `textBgColor`, `fullwidthBgColor`, `panelBgColor` and `slideColor`. We set a blue background for the text screen and the fullwidth screen, we use a bright color for the text (`slideColor`).

```
\selectColors
{
  textBgColor = blue,
  fullwidthBgColor = blue,
  panelBgColor = grey,
  slideColor = yellow
}
\end{slide}
```

- Using `\selectColors` in the preamble, sets the colors for the whole document. If used between slides, the changes are global, and take effect starting with the next slide. If used inside a slide, they are in effect only in that slide.

The web package comes with extensive support for inserting graphics, called *templates*, into the background of a page. Use of templates is fully documented in the [AcroTeX eDucation Bundle Documentation](#). We present here only the essentials.

The templates are stacked from lowest to highest in the order in which they appear in the document. The graphics at the lowest level are inserted by the `\template`, `\paneltemplate` and `\fullwidthtemplate` commands. The graphics are stretched to fit the entire screen region, whether it will be the text screen, the panel screen, or the full width screen.

```
\template[⟨includegraphics-opts⟩]{⟨path-to-template⟩}
\paneltemplate[⟨includegraphics-opts⟩]{⟨path-to-template⟩}
\fullwidthtemplate[⟨includegraphics-opts⟩]{⟨path-to-template⟩}
```

Command Location: These commands must appear in the preamble or between slides.

Parameter Description: The optional argument [`⟨includegraphics opts⟩`] can be used to pass options to the underlying `\includegraphics` command. Do not use the `width` and `height` keys; probably, it is safest to use only the Boolean keys. The Boolean key `hiresbb` is quite useful when stretching a small graphic to fit over a larger area. The one required argument is the path to the graphic file.

ex4 Example 21: Here, we use some backgrounds that are shipped with the APB to place a graphic background on the text screen and on the panel.

```
\template[hiresbb]{APB_theme_phobos_screen_ng}
\paneltemplate[hiresbb]{APB_theme_phobos_panel_ng}
\fullwidthtemplate[hiresbb]{APB_theme_phobos_screen_ng}
```

There is a *global mechanism* for inserting the `\includegraphics` optional arguments into `\template`, `\paneltemplate` and `\fullwidthtemplate`.

```
\addtotemplateArgs{⟨includegraphics-opts⟩}
\addtopaneltemplateArgs{⟨includegraphics-opts⟩}
\addtofullwidthtemplateArgs{⟨includegraphics-opts⟩}
\cleartemplateArgs{⟨includegraphics-opts⟩}
\clearpaneltemplateArgs{⟨includegraphics-opts⟩}
\clearfullwidthtemplateArgs{⟨includegraphics-opts⟩}
```

Use these commands to insert the additional optional arguments (beyond what is specified in the optional argument), or clearing the optional arguments. For example,

```
\addtotemplateArgs{hiresbb}
```

causes the `graphicx` package to use the high resolution bounding box for the graphics for `\template`.

- These templates can be removed by redefining them to be empty, `\template{}` or `\paneltemplate{}`; in this case, the background color is used. They can be overwritten, by defining them to be a different graphic, for example, `\template{my_other_graphic.eps}`.

On top of backgrounds created by `\template`, `\paneltemplate` and `\fullwidthpanel`, additional templates, perhaps a watermark or logo, can be placed using the commands `\AddToTemplate` and `\AddToPanelTemplate`.

```
\AddToTemplate{⟨template-name⟩}
\AddToPanelTemplate{⟨template-name⟩}
```

Command Location: These commands must appear in the preamble or between slides.

Parameter Description: These two commands take the name of the template. These types templates are defined using a command, `\<name>`, which contains code to tell `web` where to put the text or graphic, and what text or graphic to use; in this case, `<name>` will be the name of the template. The next example illustrates these concepts.

ex4 Example 22: Define a template, and use `\AddToTemplate` to insert it into the document. The definition may appear in the preamble, `\AddToTemplate{APBLogo}` may appear in the preamble or between slides (and take effect with the next slide).

```
\newcommand\APBLogo
{%
  \ifnum\arabic{page}>1\relax
    \parbox[b][\paperheight][c]{\textscreenwidth}%
    {\centering\includegraphics[scale=.5]{APB_Logo_bw15}}%
  \fi
}
\AddToTemplate{APBLogo}
```

The commands, `\paperheight`, `\textscreenwidth` and `\@panelwidth` (not shown in this example) are lengths that hold the paper height of the document, the width of the text screen, and the width of the panel, respectively.

The “AddTo” templates can be disabled or enabled using the four commands `\disableTemplate`, `\disablePanelTemplate`, `\enableTemplate` and `\enablePanelTemplate` from `web`. Each of these takes a single argument, the name of the template.

There are other template managing tools from the `web` package:

- `\ClearTextTemplate`, `\ClearPanelTemplate`, `\ClearAllTemplates`, `\ClearTextTemplateBuffer`, `\ClearPanelTemplateBuffer`
- `\saveElements`, `\saveClearElements`
- `\restoreElements`, `\disableScreens`, `\restoreScreens`,
- `\disablePanels`, `\restorePanels`

Additionally, APB defines `\ClearTemplatesOnly` and `\ClearAllTemplates`, these two are discussed in [Section 15.1](#).

ex14 Example 23: The tutorial file `apb_ex14.tex` discusses template manipulation techniques, and mentions some of the above commands.

- **The APB themes**

One of the most exciting features of the APB is its *themes*. A theme is a pre-designed layout of professionally designed background graphics, with color coordinated background and text. There are nine themes, each is named after one of the nine planets in the solar system of [Planet AcroTeX!](#)⁷

See the file [thm_indx.pdf](#) for a thumbnail overview of these nine themes.

A theme can be inserted into the document by the `theme` option of APB. It is not necessary to specify a *design* as a `Web` option, as these themes will automatically select `designv`.

When a theme is chosen, a configuration file is read that defines the graphics, the background colors, and the text colors. Using one of these themes, a document author is up and running in immediately, and needs only concentrate on the content of the presentation.

The paragraph ‘[Options of APB](#)’ on page 15 gives the description of the `theme` option and includes a complete list of the themes, but these are just mercury, venus, earth, mars, jupiter, saturn, uranus, neptune and pluto.

⁷Pluto is no longer considered the ninth planet, but we include it in for sentimental reasons.

Section 9: Customizing APB

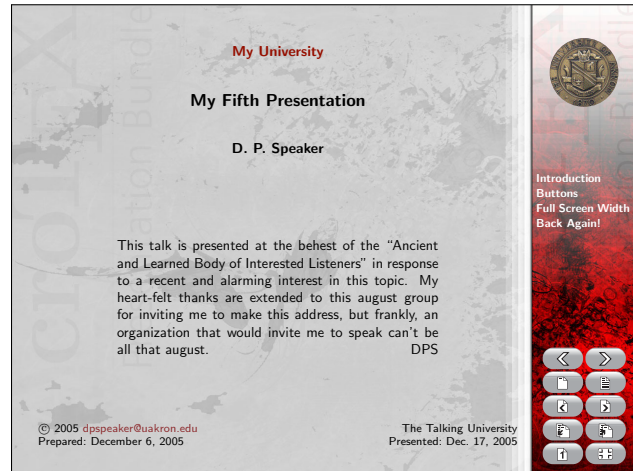
In addition to the choice of a theme, the document author can optionally select the design of the navigation buttons. There are 2D and 3D buttons to select from, each with many styles. See [‘Bottom Panel and Navibar: The Navigation Buttons’](#) on page 41.

Example 24: The following is a bit of a preamble for a APB document that illustrates usage:

```
\documentclass{article}
\usepackage[dvips,
  rightpanel]{web}
\usepackage[theme=mars]{apb}
```

The figure to the right is an image of the cover page for the mars theme.

ex5 **Example 25:** Tutorial `apb_ex5.tex` can be used for exploring the various themes, as well as other features.



• Other APB Supplied Backgrounds

The APB also comes with a number of gradient backgrounds, thirteen to be exact, that can be stretched to fit your page design (`hiresbb` recommended).

Nine of the backgrounds are based on the nine themes (and having the same planetary names), but without the watermarks and other graphic elements; the other four are new, and are named after some of the planetary moons.

These backgrounds can be used by setting the `apbBg` key, see [‘Options of APB’](#) on page 15 for a listing of the various backgrounds available. The list is given here as well for convenience: `mercury`, `venus`, `earth`, `mars`, `jupiter`, `saturn`, `uranus`, `neptune`, `pluto`, `ganymede`, `titan`, `phobos` and `europa`.

In addition to the background graphics, definitions for all relevant colors are provided through the `apbBg` option. The choices of color can be redefined using `\selectColors`, refer to [‘Selecting Colors’](#) on page 45.

ex6 **Example 26:** The following is a “typical” preamble for using the `europa` background. The demonstration file `apb_ex6.tex` shows the thirteen background graphics. It also demonstrates the use of `\useApbBg`, introduced below.

```
\documentclass{article}
\usepackage[dvips, rightpanel]{web}
\usepackage[apbBg=europa]{apb}
```

• See the file `bgs_indx.pdf` for a thumbnail overview of these thirteen backgrounds.

The initial APB background or theme should be introduced into the document via the appropriate key-value pair (`theme` or `apbBg`), the occasional person may want to change the graphic scheme one or more times throughout the document. (Can you believe it?) The next commands aid in this process.

```
\useApbBg{<apbBg-value>}
```

Command Location: Use this command between slides.

Parameter Description: The command `\useApbBg` takes as its argument any one of the values of the `apbBg` key. No checking is done. Use this command to change the background design from current one.

ex6 **Example 27:** An example of usage of `\useApbBg`:

```
\useApbBg{ganymede}
```

- **User Defined Backgrounds and Colors**

The APB comes with two files `apb_template.tex` and `template_ud.def`. The first one is a startup file for writing your own presentation. The second file can be used to create your own customized graphics and color scheme. Rename this file as `<name>_ud.def`, place it in either the folder holding your source file for your presentation, or in a folder that is in your \LaTeX search path.

The newly created background graphics and color scheme can then be input using the command `\useUdBg`. This command has the same functionality as `\useApbBg`, but it is for inputting the user-defined file `<name>_ud.def`.

```
\useUdBg{<name>}
```

Command Location: Use this command between slides.

Parameter Description: The command `\useUdBg` takes as its argument the `<name>` of a user-defined file by the name of `<name>_ud.def`.

9.4. The Navigation Panel

By default, the navigation panel is divided into three parts,⁸ each part has a corresponding macro: top (`\toppanel`), middle (`\middlepanel`) and bottom (`\bottompanel`). Each of these three commands can be redefined, see the documentation contained in `apb.dtx`, but for normal use, redefining is not necessary.

- **Top Panel: `\insertLogo`**

The top section of the panel is used for inserting a school or company logo (graphic or text); there is no restriction on usage, however. The height of the top panel is `.25\paperheight`.

The recommended method of writing to the top panel is through the `\insertLogo` command:

```
\insertLogo{<text or graphics>}
```

The material is placed into a `\parbox` with width `\@panelwidth` (default is 1 in.) and height equal to `.25\paperheight`. The graphic or text needs to be scaled to fit into this box.

ex6 **Example 28:** Example illustrating the usage of `\insertLogo`:

```
\insertLogo{\includegraphics{APB_Logo}}
```

- **Middle Panel: `paneloutline` and `\altmiddlepanel`**

The middle panel (`\middlepanel` in `apb.dtx`) contains either the outline of the presentation (a mini-table of contents) or alternate text or graphic.

If the `middlepanel` of APB is taken, the middle portion of the panel is used for an outline of the presentation, *only the section titles* appear in this outline. Relevant colors for the outline are `tocBgActive`, `tocTxtActive` and `tocTxtInactive`. The colors can be set through the `\selectColors`, see ‘[Selecting Colors](#)’ on page 45; there you will find a brief description of each of these color keys.

To adjust the formatting of the paneloutline, use `\apbpaneloutlineFormat`.

⁸Much like Gaul was divided into three parts by Caesar.


```
\apbpaneloutlineFormat{<format-commands>}
```

Command Location: Use this command in the preamble.

The default definition given is

```
\apbpaneloutlineFormat{\bfseries\footnotesize}
```

When the `\paneloutline` option is not taken, what do we do with this middle space? Answer: We write to it using `\altmiddlepanel`

```
\altmiddlepanel{<text or graphics>}
```

The material `<text or graphics>` is placed in a `\parbox` of width `\@panelwidth` and height equal to `.35\paperheight`

Command Location: This command should appear in the preamble or between slides. If placed between slides, the message in the middle panel can be changed.

- **Bottom Panel and Navibar: The Navigation Buttons**

Because APB uses layers to create talking points and other special effects, such as animations, special controls are needed to work through the document. The controls come in the form of ten buttons (twelve, actually) which when pressed, activates JavaScript code that performs the designed task, let's hope.

Table 2 lists the navigation icons along with a short description of their use. When the `slideshow` option is used, two of the buttons on the first page are replaced by the FS Button and the Slideshow Button for going into full screen mode and for starting the slide show.

- There seems to be a large number of buttons, and there are, but the most frequently pressed button is the 'Next' button. The 'Next' button makes the next layer visible, this is the button you press to work your way through the presentation. The button will change pages when the last layer is reached on the page. So just keep pressing 'Next', nothing can go wrong!

There are two types of control buttons:

1. The buttons in the Navibar, this is a row of buttons appearing at the bottom of the screen. The Navibar is used when a panel option is *not taken* or when the `fullwidth` slide option is taken. Use the `navibar` option of `web` to make the Navibar appear; it automatically appears when the `fullwidth` option of the `slide` environment is taken.
2. When a panel option is taken, a standard group of navigation buttons appear in the bottom of the navigation panel.

For the buttons that appear in the navigation panel, there is a choice of two types of buttons, 2D or 3D.

- APB has two options `use2D` and `use3D` to select either 2D (flat) buttons, or 3D buttons. `use2D` is the default and is in effect if neither option is taken.

Beyond the selection of the type of button, there are other considerations as well, such as button style, text and background colors. Use `\naviLayoutiiD` and `\naviLayoutiiiD` to further refine the design of the buttons.

```
\naviLayoutiiD{<KV-pairs>}
```

When the `use2D` option is taken, this command is used to design the colored boxes and dings that make up the navigation buttons. The `use2D` gives the smallest file sizes.

Command Location: This command can appear in the preamble, or between slides.

Key-Value Pairs: The argument takes key-value pairs to design the buttons.

1. `style`: Permissible values: `flat01`, `flat02`, `flat03`, `flat04` and `flat05`. The default is `flat01`.
2. `textColor`: Any defined color, the default is `webblue` (defined in `Web`).
3. `bgColor`: Any defined color, the default is `webgrey` (defined in `Web`).

• The icons of `flat05` are created from graphics files, hence, they yield larger file sizes than the other style parameter values. Styles `flat01`–`flat04` use various characters from common \LaTeX font sets.

Example 29: We set the argument for `\naviLayoutIID`.

```
\naviLayoutIID{style=flat01,bgColor=red,textColor=green}
```

When the `use3D` option is taken, this command is used to design the three-dimensional buttons used for navigation.

```
\naviLayoutIIID{<KV-pairs>}
```

The buttons and icons are graphics files designed by a graphics professional. The `use3D` gives larger file sizes than the 2D option.

Command Location: This command can appear in the preamble, or between slides.

Key-Value Pairs: The argument takes key-value pairs to design the buttons.

1. `style`: Permissible values are `big01`, `big02`, `big03`, `big04`, `rect01`, `rect02`, `round01`, `round02`, `round03` and `round04`. The default is `big01`.
2. `color`: Permissible values are `black`, `blue`, `green`, `grey`, `red` and `yellow` defined color. The default is `grey`.
3. `bgColor`: Any defined color, the default is transparent.

• The navigation icons that are used for the 3D buttons are shown in [Table 2](#), they correspond to the `flat05` set.

Example 30: We set the argument of `\naviLayoutIIID`.

```
\naviLayoutIIID{style=round03,color=grey,bgColor=webyellow}
```

There is an additional command that effects the navigation buttons and links.

```
\highlightType{<N|I|O|P>}
```

The argument of the `\highlightType` command sets the type of highlighting for the navigation icons. Permissible values are `N` (no highlighting), `I` (invert, the default), `O` (Outline) and `P` (Push, also called Inset in the **Acrobat** UI).

Example 31: This sets the highlight style to push (or inset).

```
\highlightType{P}
```

It is possible to change between 2D and 3D between slides as well.

```
\useIID \useIIID
```

Use the `\useIID` and/or the `\useIIID` between slides to switch over between button styles. You can use `\naviLayoutIID` and `\naviLayoutIIID` between slides to redesign the button styles.

`\NavbarOnFirstPage`

When the `navbar` of `web` is taken, the navigation bar appears at the bottom of the page (assuming a `panel` option is not taken). By default, it does not appear on the first page, which is usually the title page. If you have no title page, you can force the navigation bar to appear on the first page with this command. Place this command in the preamble.

Name	Description	Flat01	Flat02	Flat03	Flat04	Flat05
Prev	navigates to the previous layer, if previous layer is on the previous page, will go to the previous page.	◀	◀	◀	◀	◀
Next	navigates to the next layer, if next layer is on the next page, will go to the next page.	▶	▶	▶	▶	▶
PrevPage	go to the previous page.	◀	◀	◀	◀	◀
NextPage	go to next page.	▶	▶	▶	▶	▶
TopOfPage	hides all layers on current page, makes current layer the layer at the top of the page.	▲	▲	▲	▲	▲
BotOfPage	makes visible all layers on the current page, makes the current layer the one at the bottom of the page.	▼	▼	▼	▼	▼
TopOfPagePrev	A combination of both <code>TopOfPage</code> and <code>PrevPage</code> .	◀	◀	◀	◀	◀
BotOfPageNext	A combination of both <code>BotOfPage</code> and <code>NextPage</code> .	▶	▶	▶	▶	▶
FirstPage	makes all layers hidden, and goes to the first page.	◀	◀	◀	◀	◀
Close	leave full screen mode, or close document.	■	■	■	■	■
FS	go into full screen mode.	□	□	□	□	□
SlideShow	start slideshow.	□	□	□	□	□

Table 2: List of Navigation Icons with Descriptions

9.5. Set Default Slide Options

The `slide` environment can be customized in several different of ways through the use of the command `\setDefaultSlides`, a description follows.

`\setDefaultSlides{{KV-pairs}}`

The command sets the default options for the slide environment. These values can be overwritten by resetting the key-value pairs as part of the optional argument of the `slide` environment.

Command Location: Use this command in the preamble or between slides.

Key-Value Pairs: The argument takes key-value pairs to design the buttons.

1. `narrower`: A Boolean, which if true, narrows the margins of the slide, perhaps for a more eye-pleasing presentation.
2. `indent`: A length that is used by `narrower` increase the left and right margins. The default value is `\apbIndent`, which on startup is set to 20pt.
3. `vcenter`: A Boolean, which if true, attempts to center the content of the slide vertically. The normal behavior is not to center.

Section 9: Customizing APB

4. `fontsize`: The default size of the font to appear within this slide. Permissible values are `tiny`, `scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `LARGE`, `huge` and `Huge`. These correspond to the standard font sizes. The default is the class default size.
5. `fontseries`: Sets the default series for this slide. Permissible values are `bfseries` and `mdseries`. The default is the class default font series.
6. `color`: Sets the color of text for this slide. Any name of any named color is permissible. The default is T_EX's default color, usually black.

Example 32: The declaration sets the defaults for all slides in this presentation, appropriate, perhaps, for a dark background.

```
\setDefaultSlides{fontsize=large,fontseries=bfseries,color=yellow}
```

9.6. Selecting Dings and Things

In this subsection, we discuss the selection of dings for the talking point command `\dPt`, `\ddPt` and `\dddPt` and for the dings that appear in the table of contents. We also discuss selection of colors.

• Selecting Dings

This section discusses how to select the dings that appear at the beginning of the commands `\dPt`, `\ddPt` and `\dddPt` and in the table of contents listing.

```
\selectDings{<KV-pairs>}
```

Command Location: Place anywhere, within a slide environment, changes are only local.

Key-Value Pairs: `\selectDings` takes several key-value pairs, all having default values, so not all need appear. In all cases below, to have no ding, set a key equal to `\noDing`.

1. `dDing`: The ding used for `\dPt`. (APB 2.0) If you set `dDing` equal to `enum`, then you get a numbered ding: 1, 2, 3, etc.
2. `ddDing`: The ding used for `\ddPt`. (APB 2.0) If you set `ddDing` equal to `enum`, then you get a lettered ding: (a), (b), (c), etc.
3. `dddDing`: The ding used for `\dddPt`. (APB 2.0) If you set `dddDing` equal to `enum`, then you get a ding with roman numerals: i, ii, iii, etc.
4. `dDingToc`: The ding used for `\dPt` in table of contents. (See [Section 6.2](#))
5. `ddDingToc`: The ding used for `\ddPt` in table of contents.
6. `dddDingToc`: The ding used for `\dddPt` in table of contents.

Example 33: This example illustrates the use of `\selectDings`,

```
\selectDings
{
  dDing=\ding{080},
  ddDing=\ding{042},
  dddDing=\ding{171},
  dDingToc=$\clubsuit$,
  ddDingToc=$\spadesuit$,
  dddDingToc=$\bigstar$
}
```

Section 9: Customizing APB

Should you wish to create an enumerated list of talking points give `dDing`, `ddDing` or `dddDing` keys a special value of `enum`, as in the next example. You can mix in the special value of `enum` with other definitions, such as `\ding{042}`.

Example 34: Enumerate the talking points:

```
\selectDings{dDing=enum,ddDing=enum,dddDing=enum}
```

APB uses a labeling system similar to that of the `enumerate` environment, as [Table 3](#) shows. The default definitions of the `\theapbcni`, `\theapbcnii` and `\theapbcniii` counters can be redefined as can the default forms of the labels `\labelapbcnti`, `\labelapbcntii` and `\labelapbcntiii`.

Example 35: Redefine the first level label so the number is in bold.

```
\renewcommand{\labelapbcnti}{\textbf{\theapbcnti.}}
```

	First Level	Second Level	Third Level
<i>counter</i>	<code>apbcnti</code>	<code>apbcntii</code>	<code>apbcntiii</code>
<i>representation</i>	<code>\theapbcnti</code>	<code>\theapbcntii</code>	<code>\theapbcntiii</code>
<i>default definition</i>	<code>\arabic{apbcnti}</code>	<code>\alph{apbcntii}</code>	<code>\roman{apbcntiii}</code>
<i>label field</i>	<code>\labelapbcnti</code>	<code>\labelapbcntii</code>	<code>\labelapbcntiii</code>
<i>default form</i>	<code>\theapbcnti.</code>	<code>(\theapbcntii)</code>	<code>\theapbcntiii.</code>

Table 3: Commands controlling the `enum` value

The counter `apbcnti` enumerates the first level talking points and is not automatically reset. To reset it—that is, to start the count over at 1 again—use `\resetAPBEnum`:

```
\resetAPBEnum
```

Command Location: Place anywhere you want the counters to be reset.

- There are command versions of these settings. Use them if you prefer. They are `\dDing{<ding>}`, `\ddDing{<ding>}` and `\dddDing{<ding>}`. The default definitions of this set are given as follows: `\dDing{\ding{079}}`, `\ddDing{\ding{169}}` and `\dddDing{\Large\bullet}`.

• Selecting Colors

```
\selectColors{<KV-pairs>}
```

Through this command, you set the color of most all elements of the document that have a color attribute.

Command Location: There are no restrictions on this command. In the preamble, it will set the defaults for the entire document. Between slides, it will effect the document beginning with the next slide. Within a slide, the definitions will only be local to that slide.

Key-Value Pairs: Each of the following keys take a named color. Many of these keys have a command interface as well, these are also noted.

- `universityColor`: The color of the value of the `\university` declaration, see [page 18](#). The value of `\university` appears on the title page. The default is blue.
- `titleColor`: The color of the value of the `\title` declaration (see [page 18](#)). The value of `\title` appears on the title page. The default is black.

Section 9: Customizing APB

3. `authorColor`: The color of the value of the `\author` declaration (see page 18). The value of `\author` appears on the title page. The default is black.
4. `tocBgActive`: When the `paneloutline` option is taken, this is the color of the background of the highlighted item. The default is blue.
5. `tocTxtActive`: When the `paneloutline` option is taken, this is the color of the text of the highlighted item. The default is red.
6. `tocTxtInactive`: When the `paneloutline` option is taken, this is the normal color of an item in the outline list. The default is red.
7. `textBgColor`: The background color of the text screen. The default is white.
8. `panelBgColor`: The background color of the navigation panel. The default is white.
9. `secColor`: The color of the section heading. The default is blue.
10. `ssecColor`: The color of the subsection heading. The default is blue.
11. `sssecColor`: The color of the subsubsection heading. The default is blue.
12. `fullwidthBgColor`: The background color of a slide that uses the `fullwidth` option. The default is white.
13. `slideColor`: The color of all text appearing in a slide. The default is black.
14. `dBldColor`: The color of text created by the `\dBld` command. Default is black.
15. `dPtColor`: The color of text created by the `\dPt` command. Default is red. The command version is `\dPtColor`, e.g., `\dPtColor{red}`.
16. `ddPtColor`: The color of text created by the `\ddPt` command. Default is webbrown. The command version is `\ddPtColor`, e.g., `\ddPtColor{webbrown}`.
17. `dddPtColor`: The color of text created by the `\dddPt` command. Default is webblue. The command version is `\dddPtColor`, e.g., `\dddPtColor{webblue}`.
18. `dDingColor`: The color of the ding for the `\dPt` command. Default is red. The command version is `\dDingColor`, e.g., `\dDingColor{red}`.
19. `ddDingColor`: The color of the ding for the `\ddPt` command. Default is webbrown. The command version is `\ddDingColor`, e.g., `\ddDingColor{webbrown}`.
20. `dddDingColor`: The color of the ding for the `\dddPt` command. Default is webblue. The command version is `\dddDingColor`, e.g., `\dddDingColor{webblue}`.
21. `urlColor`: The color of an URL link. The default is webbrown.
22. `linkColor`: The color of a link. The default is webgreen.
23. `fileColor`: The color of a link to a local file. The default is webbrown.

There are several other colors, these can be set through their interface, see, [Section 9.7](#) for a discussion of `\sectionLayout`, `\subsectionLayout` and `\subsubsectionLayout`.

9.7. Section Headings

We use the usual section headings `\section`, `\subsection` and `\subsubsection`, but they have been modified so that the document author can easily design how the section titles look.

```
\useSectionNumbers
```

The command `\useSectionNumbers` causes APB to generate the usual numbering system for a \LaTeX document. The default is to use no section numbers, seems appropriate for a presentation.

Command Location: The use of this command is restricted to the preamble. The decision must be made for the whole document at the beginning of the document.

```
\sectionLayout{<KV-pairs>}  
\subsectionLayout{<KV-pairs>}  
\subsubsectionLayout{<KV-pairs>}
```

Command Location: No restriction on the use of this command, in the preamble, or between slides changes are global; inside a slide environment, changes occur within a group.

Key-Value Pairs: Each of these takes the same key-value pairs.

1. `fontfamily`: Font family to use for the section title, permissible values are `rmfamily`, `sffamily`, `ttfamily`.
2. `fontseries`: Font series to use for the section title, values are `bfseries` and `mdseries`.
3. `fontshape`: Font shapes to use for the section title, values are `upshape`, `itshape`, `scshape`, `slshape`.
4. `fontsize`: Font size of the section title, permissible values are `tiny`, `scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `LARGE`, `huge`, `Huge`.
5. `halign`: Alignment of the section title, permissible values are `l` (left aligned), `c` (centered), `r` (right aligned).
6. `ding`: You can specify a ding to display where the section number is usually placed; for example, `ding=\ding{066}`.
7. `color`: The color of the section title, this can be any named color. The default is blue.
8. `special`: Through this key, you can specify predefined layout for the section titles. Permissible values: `shadow`, `framebox`, `colorbox`, `fcolorbox`, `frameboxfit`, `colorboxfit`, `fcolorboxfit`, `custom`, `default`.

Custom section heads can be used by specifying a value of `custom`. APB uses the commands `\customSecHead`, `\customSubsecHead` and `\customSubsubsecHead`. These are macros that take one argument, the code for designing the title. The title is referred to as `#1`. Depending on how these custom section titles are defined, the other keys may not be obeyed. See an [Example 37](#) below; there are other examples in the demo files.

9. `framecolor`: The color of the frame surrounding the subject when the `special` key has a value of `framebox`, `fcolorbox`, `frameboxfit` or `fcolorboxfit`.
10. `bgcolor`: The background (fill color) of the box enclosing the section title, when `special` has a value of `colorbox`, `fcolorbox`, `colorboxfit` or `fcolorboxfit`.
11. `shadowcolor`: The color of the shadow, when `special` has a value of `shadow`.
12. `beforeskip`: The amount of skip before the section title.

13. `afterskip`: The amount of skip after the section title.
14. `usefont`: Through this key it is possible to specify an arbitrary font and font size. The key takes five parameters, for example, `usefont={OT1}{cmdh}{m}{n}{{16}{16pt}}`. The first four are the arguments of the \TeX 's `\usefont`, encoding, family, series and shape. The last argument are the arguments of the \TeX 's `\fontsize`, size and baselineskip.
If the fifth parameter is empty, no font size is specified, the current default sizes are used, e.g., `usefont={OT1}{cmdh}{m}{n}`.
15. `reset`: This key attempts to reset changes to their defaults. Permissible values are `font` (to reset the font changes only), or `all` (to reset all the keys).

ex7 Example 36: The following example sets the section title with shadows: text is red and shadow is blue. We center the title horizontally, and specify an after skip of 12pt.

```
\sectionLayout{afterskip=12pt,halign=c,color=red,shadowcolor=blue}
```

Similarly for `\subsectionLayout` and `\subsubsectionLayout`.

ex7 Use `apb_ex7.tex` to explore the section layout commands.

```
\shadowoffset{<length>}
\shadowvoffset{<length>}
```

When `special = shadow`, shadowing is obtained by typesetting the text twice with different colors. The amount of horizontal and vertical offset are controlled by these two commands. The defaults are `\shadowoffset{.2ex}` and `\shadowvoffset{-.2ex}`.

The interface for writing custom section layouts are the following three commands.

```
\customSecHead{<tex_code>}
\customSubsecHead{<tex_code>}
\customSubsubsecHead{<tex_code>}
```

Command Location: Use in the preamble or between slides.

Parameter Description: These three commands are used to write custom section, subsection and subsubsection layouts. The `<tex_code>` is \TeX code for laying out the section titles, and should use `#1` to represent the section title.

Example 37: The following definitions can appear anywhere, they are global unless appearing in a `slide` environment. This puts a colored box around the section title, which is assumed to appear in one line. In the example below, we use the internal color names for the `color` (`\apb@sectioncolor`) and `bgcolor` (`\apb@sectionbgcolor`) keys, in this way, the title will obey the values of these keys. This particular custom head obeys the `halign` key as well.

```
\customSecHead{\vbox{\colorbox{\apb@sectionbgcolor}
  {\color{\apb@sectioncolor}#1}}}
\sectionLayout{special=custom,halign=l,bgcolor=red,color=white}
```

The above code is a simplified version of the `colorboxfit`, a value of the `special` key.

ex7 See `apb_ex7.tex` for a working example of custom section head layout.

Important: It is important to remark that between slides `@` is made into a 'letter', and reset to 'other' at the beginning of the next slide. This makes it convenient to make these definitions between slides. In the preamble, however, such a definition must be between `\makeatletter` and `\makeatother`.

9.8. Running Headers and Footers

The web package provides some basic header and footer support; however, the APB uses the footers to insert the Navibar, so use of the footer is discouraged.

For the headers, web provides `\lheader`, `\cheader` and `\rheader`. The APB makes the following definitions:

```
\lheader{\apb@setmarks}
\rheader{\ifnum\value{page}>1\relax
  \thepage/\curname apbLastPage\endcurname\fi}
```

The `\lheader` calls the macro `\apb@setmarks` that sets the section title in the left header. On the page `\section` appears, the title does not appear in the running header; on subsequent pages, the title does appear, until there is a new section. The document author is free to redefine `\lheader` as desired, with or without the running section titles.

The right header, `\rheader`, provides the page number and the number of pages in the form 3/6, meaning page 3 of a total of 6 pages. You are certainly free to overwrite this definition.

The `\cheader` is not used by APB, so you are free to create a running header centered between the other two headers.

The APB provides one other command that may be of use,

```
\headerformat{<format-cmds>}
```

This command can be used to format, in a uniform way, the entire running header. For the themes package, the following definition is made

```
\headerformat{\bfseries\color{\apb@sectioncolor}}
```

The running headers will be in bold, with the same color as the section color (`\apb@sectioncolor` is the internal name for the color for the section.)

On occasion you may want to have no running headers at all, this can be accomplished by saying in the preamble, or between slides `\lheader{}`, `\cheader{}`, and/or `\rheader{}`. Another solution is to use the following commands.

```
\clearHeaders
\restoreHeaders
```

Command Location: Place these commands in the preamble or between slides.

Command Description: `\clearHeaders` first saves the current definitions of the headers then sets the headers to the empty header. `\restoreHeaders` restores the definitions that were in effect when the last `\clearHeaders` was executed.

10. Refining the Presentation

In Sections 5 through 9, the basics of building an APB presentation document were introduced. Now we discuss refining the presentation. We describe how to set full screen options (Section 10.1) for the PDF viewer, to set page transitions (Section 10.2), to create page open and page close events (Section 10.4) and to embed sounds (Section 10.3) into your PDF.

10.1. Set Fullscreen Options

The Adobe Reader (and **Acrobat** application) has a full screen mode, typically used for presentations, such as talks (full screen does not work when the PDF is in a web browser). There are user preferences for setting the default behavior of the viewer while in full screen mode. The full screen preferences are accessed through the menu

Section 10: Refining the Presentation

Edit > Preferences > General > Full Screen

The shortcut (on Windows) to the preferences dialog is Ctrl+K. Most of the fields seen there can be controlled programmatically through JavaScript. The APB's access to these JavaScript commands is through the `\setDefaultFS` command.

```
\setDefaultFS{<KV-pairs>}
```

The command for setting how you want the viewer to behave in fullscreen. This command is implemented through JavaScript, as opposed to the `pdfmark` operator. See *Acrobat JavaScript Scripting Reference* [1], the section on the `FullScreen` object.

Command Location: This command must be executed in the preamble.

Key-Value Pairs: The command has numerous key-value pairs, the defaults of most of these are set in the Preferences menu of the viewer. These values are the ones listed in the *Acrobat JavaScript Scripting Reference* [1].

1. **Trans:** permissible values are `NoTransition`, `UncoverLeft`, `UncoverRight`, `UncoverDown`, `UncoverUp`, `UncoverLeftDown`, `UncoverLeftUp`, `UncoverRightDown`, `UncoverRightUp`, `CoverLeft`, `CoverRight`, `CoverDown`, `CoverUp`, `CoverLeftDown`, `CoverLeftUp`, `CoverRightDown`, `CoverRightUp`, `PushLeft`, `PushRight`, `PushDown`, `PushUp`, `PushLeftDown`, `PushLeftUp`, `PushRightDown`, `PushRightUp`, `FlyInRight`, `FlyInLeft`, `FlyInDown`, `FlyInUp`, `FlyOutRight`, `FlyOutLeft`, `FlyOutDown`, `FlyOutUp`, `FlyIn`, `FlyOut`, `Blend`, `Fade`, `Random`, `Dissolve`, `GlitterRight`, `GlitterDown`, `GlitterRightDown`, `BoxIn`, `BoxOut`, `BlindsHorizontal`, `BlindsVertical`, `SplitHorizontalIn`, `SplitHorizontalOut`, `SplitVerticalIn`, `SplitVerticalOut`, `WipeLeft`, `WipeRight`, `WipeDown`, `WipeUp`, `WipeLeftDown`, `WipeLeftUp`, `WipeRightDown`, `WipeRightUp`, `Replace`, `ZoomInDown`, `ZoomInLeft`, `ZoomInLeftDown`, `ZoomInLeftUp`, `ZoomInRight`, `ZoomInRightDown`, `ZoomInRightUp`, `ZoomInUp`, `ZoomOutDown`, `ZoomOutLeft`, `ZoomOutLeftDown`, `ZoomOutLeftUp`, `ZoomOutRight`, `ZoomOutRightDown`, `ZoomOutRightUp`, `ZoomOutUp`, `CombHorizontal`, `CombVertical`. The default is `Replace`.

(APB 2.0) The following are new to **Acrobat/Adobe Reader** version 8: `PushLeftDown`, `PushLeftUp`, `PushRightDown`, `PushRightUp`, `WipeLeftDown`, `WipeLeftUp`, `WipeRightDown`, `WipeRightUp`, `ZoomInDown`, `ZoomInLeft`, `ZoomInLeftDown`, `ZoomInLeftUp`, `ZoomInRight`, `ZoomInRightDown`, `ZoomInRightUp`, `ZoomInUp`, `ZoomOutDown`, `ZoomOutLeft`, `ZoomOutLeftDown`, `ZoomOutLeftUp`, `ZoomOutRight`, `ZoomOutRightDown`, `ZoomOutRightUp`, `ZoomOutUp`, `CombHorizontal`, `CombVertical`

The transition chosen by this key will be in effect for each page that does not have a transition effect separately defined for it (by the `\setPageTransition` command).

2. **bgColor:** Sets the background color in fullscreen mode. The color specified must be a JavaScript Color array, for example, `bgColor=["RGB" 0 1 0]`, or you can use one of the preset colors, like `bgColor=color.ltGray`.
3. **timeDelay:** The default number of seconds before the page automatically advances in full screen mode. See `useTimer` to activate/deactivate automatic page turning.
4. **useTimer:** A Boolean that determines whether automatic page turning is enabled in full screen mode. Use `timeDelay` to set the default time interval before proceeding to the next page.
5. **loop:** A Boolean that determines whether the document will loop around back to the first page.
6. **cursor:** Determines the behavior of the mouse in full screen mode. Permissible values are `hidden`, `delay` (hidden after a short delay) and `visible`.
7. **escape:** A Boolean use to determine if the escape key will cause the viewer to leave full screen mode.

Section 10: Refining the Presentation

8. `clickAdv`: A Boolean that determines whether a mouse click on the page will cause the page to advance.
9. `fullscreen`: A Boolean, which if `true`, causes the viewer to go into full screen mode. Has no effect from within a browser.
10. `usePageTiming`: A Boolean that determines whether automatic page turning will respect the values specified for individual pages in full screen mode (which can be set through `\setDefaultFS`).

Example 38: This example causes the viewer to go into full screen mode, sets the transition to `Random`, instructs the viewer to loop back around to the first page, and to make the cursor hidden after a short period of inactivity.

```
\setDefaultFS{fullscreen,Trans=Random,loop,cursor=delay,escape}
```

- On closing the document, the user's original full screen preferences are restored.

10.2. Set Page Transitions

The `\setDefaultFS` command can set the full screen behavior of the viewer for the *entire document*, including a transition effect applicable to all pages in the document; for transition effects of individual pages, use the `\setPageTransition` command.

```
\setPageTransition{(KV-pairs)}
```

Sets the transition effect for the *next page only*, viewer must be in full screen mode. The command `\setPageTransition` is implemented using the `pdfmark` operator.

Command Location: This command should be used in the preamble for the first page, and between slides for subsequent pages.

Key-Value Pairs: The `\setPageTransition` command has several key-value pairs:

1. `Trans`: permissible values are `NoTransition`, `UncoverLeft`, `UncoverRight`, `UncoverDown`, `UncoverUp`, `UncoverLeftDown`, `UncoverLeftUp`, `UncoverRightDown`, `UncoverRightUp`, `CoverLeft`, `CoverRight`, `CoverDown`, `CoverUp`, `CoverLeftDown`, `CoverLeftUp`, `CoverRightDown`, `CoverRightUp`, `PushLeft`, `PushRight`, `PushDown`, `PushUp`, `PushLeftDown`, `PushLeftUp`, `PushRightDown`, `PushRightUp`, `FlyInRight`, `FlyInLeft`, `FlyInDown`, `FlyInUp`, `FlyOutRight`, `FlyOutLeft`, `FlyOutDown`, `FlyOutUp`, `FlyIn`, `FlyOut`, `Blend`, `Fade`, `Random`, `Dissolve`, `GlitterRight`, `GlitterDown`, `GlitterRightDown`, `BoxIn`, `BoxOut`, `BlindsHorizontal`, `BlindsVertical`, `SplitHorizontalIn`, `SplitHorizontalOut`, `SplitVerticalIn`, `SplitVerticalOut`, `WipeLeft`, `WipeRight`, `WipeDown`, `WipeUp`, `WipeLeftDown`, `WipeLeftUp`, `WipeRightDown`, `WipeRightUp`, `Replace`, `ZoomInDown`, `ZoomInLeft`, `ZoomInLeftDown`, `ZoomInLeftUp`, `ZoomInRight`, `ZoomInRightDown`, `ZoomInRightUp`, `ZoomInUp`, `ZoomOutDown`, `ZoomOutLeft`, `ZoomOutLeftDown`, `ZoomOutLeftUp`, `ZoomOutRight`, `ZoomOutRightDown`, `ZoomOutRightUp`, `ZoomOutUp`, `CombHorizontal`, `CombVertical`. The default is `Replace`.

(APB 2.0) The following are new to **Acrobat/Adobe Reader** version 8: `PushLeftDown`, `PushLeftUp`, `PushRightDown`, `PushRightUp`, `WipeLeftDown`, `WipeLeftUp`, `WipeRightDown`, `WipeRightUp`, `ZoomInDown`, `ZoomInLeft`, `ZoomInLeftDown`, `ZoomInLeftUp`, `ZoomInRight`, `ZoomInRightDown`, `ZoomInRightUp`, `ZoomInUp`, `ZoomOutDown`, `ZoomOutLeft`, `ZoomOutLeftDown`, `ZoomOutLeftUp`, `ZoomOutRight`, `ZoomOutRightDown`, `ZoomOutRightUp`, `ZoomOutUp`, `CombHorizontal`, `CombVertical`

These values are the ones listed in the *Acrobat JavaScript Scripting Reference* [1].

2. `TransDur`: Duration of the transition effect, in seconds. Default value: 1.

Section 10: Refining the Presentation

3. **Speed:** (APB 2.0) Same as **TransDur**, the duration of the transition effect, except this key takes values **Slow**, **Medium** or **Fast**, corresponding to the **Acrobat** UI. If **TransDur** and **Speed** are both specified, **Speed** is used. Use **TransDur** for finer granularity.
4. **PageDur:** The *PDF Reference, version 1.6* [4], describes this as “The page’s display duration (also called its advance timing): the maximum length of time, in seconds, that the page is displayed during presentations before the viewer application automatically advances to the next page. By default, the viewer does not advance automatically.”

Example 39: We set the page transition, with the following parameters.

```
\setPageTransition{Trans=Blend,PageDur=20,TransDur=5}
```

- It is possible to set the transition effects for multiple pages by using the JavaScript method `setPageTransitions`. This method is only available in the **Acrobat** application, but once executed, the effects of the method can be saved in the document. The document can then be read by the Adobe Reader.

10.3. Importing Sounds

You can play a sound (.wav) clip when a certain layer is made visible or when a particular page is opened. First, you have to declare the sounds to be embedded in the document.

```
\importSounds[<path>]{<sound-files>}
```

The first parameter, *<path>*, is the common path to all the sound files, this must be a **DIPath** (device independent path, as defined by the *PDF Reference*). The second parameter is a comma-delimited listing of sound clips to be imported into the document.

Command Location: This command *must be in the preamble*, just above the beginning of the document body.

Example 40: The first line imports two sounds clips in the same folder as the source document, the second line specifies the same two clips, but they are now located in the subfolder `sounds` of the source file folder. The last line gives an absolute reference to the location of the sounds using a **DIPath** reference.

```
\importSounds{clap.wav,trek.wav}  
\importSounds[sounds]{clap.wav,trek.wav}  
\importSounds[/c/acrotex/apb/sounds]{clap.wav,trek.wav}
```

Once the files are imported into the **PDF** document, they can be played by some of the build commands, or you can create an open page action that plays one of the sound clips.

- Only one `\importSounds` command can be used per document. If multiple commands are used, only the last one will have any effect.

Once the sounds are successfully embedded in the document, they can be played in a number of different ways:

1. A push button or link can play a sound clip
2. An open or close action can play a sound clip
3. A layer can activate a sound as it becomes visible (or hidden)

The example that follows illustrates (1); the second item is demonstrated in [Section 10.4, Example 42](#); activating a sound using a layer event is covered in the discussion of `\fbld` on page 59 and of `\bld` on page 61. See also `apb_sound.tex` and `apb_demo.tex`.

Example 41: This example illustrates the construction of a button and link for playing a sound. We assume that `\importSounds{clap.wav,trek.wav}` appears in the preamble of the document. We use the `eforms` package to create a button and a link.

```
\pushButton[\CA{Play!}  
  \A{\JS{this.getSound("trek.wav").play();}}]{trekSound}{24bp}{11bp}  
  
\setLinkText[\A{\JS{this.getSound("clap.wav").play();}}]{Clap!}
```

The code uses `getSound` to retrieve the sound object for the sound clip; `play()` is a method of this object. See *Acrobat JavaScript Scripting Reference* [1] for details of these objects and methods.

10.4. The `addJSToPageOpen`/`addJSToPageClose` Environments

When a page opens or closes in the viewer, a *page event* occurs, as Adobe terms it. It is possible to associate a JavaScript action for any page event.

APB already has a page open action defined for each page; this action sets the current layer on the page, this is why you can move around in the document, return to a page and continue where you left off on the page. The document author can define additional open page (and close page) JavaScript actions through the `addJSToPageOpen` and `addJSToPageClose` Environments.

```
\begin{addJSToPageOpen}  
  \Acrobat JavaScript  
\end{addJSToPageOpen}  
  
\begin{addJSToPageClose}  
  \Acrobat JavaScript  
\end{addJSToPageClose}
```

Environment Location: Use these environments between slides, effective the next slide. The first slide (possibly generated by `\maketitle`) is a special case: *When setting the open and close page actions for the first page, these environments must go in the preamble.*

Example 42: The following example shows how to use `addJSToPageOpen` to play a sound exactly once. The code assumes the sound clip has been imported by `\importSounds`.

```
\begin{addJSToPageOpen}  
if ( typeof playTrekSound == "undefined" ) {  
  this.getSound("trek.wav").play();  
  var playTrekSound=true;  
}  
\end{addJSToPageOpen}
```

10.5. Special Acrobat Techniques

- **Setting Transitions for Multiple Pages**

Using `\setPageTransition` to set transition effects of multiple pages may be impractical. In this case, you can set the effects of multiple pages using the JavaScript method `setPageTransitions`.

This function needs only be executed once, after distillation. Place this command in an `execJS` environment (see the `insdljs` package documentation), like so,

Section 11: Declaring the Initial View

```
\begin{execJS}{execjs}
this.setPageTransitions({
  nStart: 2,
  nEnd: 6,
  aTrans: [-1, "Random", 1]
});
\end{execJS}
```

See the *Acrobat JavaScript Scripting Reference* [1] for details.

The other possibility is to execute the above script in the **Acrobat** Debugger Console. Click on Ctrl+J, copy and paste your `setPageTransitions` code in the window, highlight it with the mouse, then execute it by pressing Ctrl+Enter (or just the Enter key on the numeric keypad).

- **Locking Layers**

The layers can be programmatically locked, which prevents anyone using the UI to manipulate the layers in the **Layers Navigation Tab** of the viewer. This **Layers Navigation Tab** can be viewed by going to the menu item View > Navigation Tabs > Layers.

In the preamble, place the following code:

```
\begin{execJS}{lockOCGs}
var ocg = this.getOCGs();
for ( var i=0; i<ocg.length; i++) ocg[i].locked=true;
\end{execJS}
```

☛ Though the layers are locked, they can still be manipulated using JavaScript.

- **Auto Saving**

After the distill is completed, normally the document opens in **Acrobat**. At this point, there may be some JavaScript being executed, such as importing document level JavaScript, importing sounds or locking the layers. After everything settles down, the document *must be saved*, this permanently embeds the newly imported JS and sounds in the document.

☛ **Important:** After distillation, *the document must always be saved*.

To automate the task of always saving, you can place in the preamble the following code:

```
\begin{execJS}{execjs}
aebTrustedFunctions(this,aebSaveAs,"Save");
\end{execJS}
```

This code should be placed last in the queue of all `execJS` environments. Saving earlier effectively stops the execution of later code in an `execJS` environments.

You can, of course, combine several tasks into one `execJS` environment, as follows.

```
\begin{execJS}{lockSave}
var ocg = this.getOCGs();
for ( var i=0; i<ocg.length; i++) ocg[i].locked=true;
aebTrustedFunctions(this,aebSaveAs,"Save");
\end{execJS}
```

Notice that `aebTrustedFunctions(this,aebSaveAs,"Save")` is the last line of the script.

Section 11: Declaring the Initial View

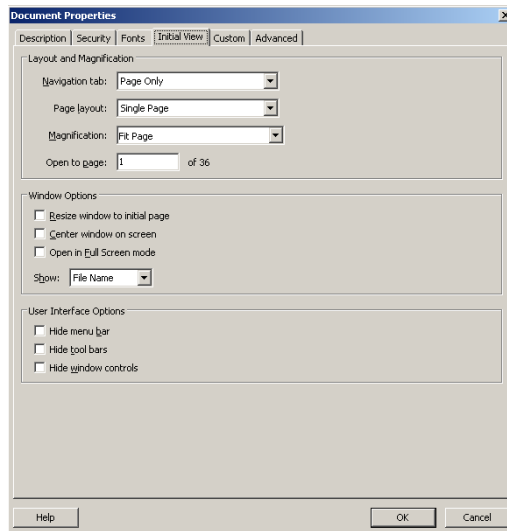


Figure 6: Initial View of Document Properties

11. Declaring the Initial View

`\DeclareInitView` is a “data structure” for setting the Initial View of the Document Properties dialog box, See [Figure 6](#).

`\DeclareInitView` takes up to three key-value pairs, the three keys correspond to the three named regions of the UI (User Interface):

Key	User Interface Name
<code>layoutmag</code>	Layout and Magnification
<code>windowoptions</code>	Window Options
<code>uioptions</code>	User Interface Options

The values of each these three are described in the tables below:

- `layoutmag`: This key sets the initial page layout and magnification of the document. The values of this key are themselves key-values:

Key	Value(s)	Description
<code>navitab</code>	<code>UseNone</code> , <code>UseOutlines</code> , <code>UseThumbs</code> , <code>UseOC</code> , <code>UseAttachments</code>	The UI for these are: Page Only, Bookmarks Panel and Page, Pages Panel and Page, Layers Panel and Page, Attachments Panel and Page, respectively. The default is <code>UseNone</code>
<code>pagelayout</code>	<code>SinglePage</code> , <code>OneColumn</code> , <code>TwoPageLeft</code> , <code>TwoColumnLeft</code> , <code>TwoPageRight</code> , <code>TwoColumnRight</code>	The UI for these are: Single Page, Single Page Continuous, Two-Up (Facing), Two-Up Continuous (Facing), Two-Up (Cover Page), Two-Up Continuous (Cover Page), respectively. The default is user’s preferences.
<code>mag</code>	<code>ActualSize</code> , <code>FitPage</code> , <code>FitWidth</code> , <code>FitHeight</code> , <code>FitVisible</code> , or <i>(positive number)</i>	The UI for these are: Actual Size, Fit Page, Fit Width, Fit Height, Fit Visible, respectively. If a positive number is provided, this is interpreted as a magnification percentage. The default is to use user’s preferences.
<code>openatpage</code>	<i>(positive number)</i>	The page number (base 1) to open the document at. Default is page 1.

- `windowoptions`: The Window Options region of the Initial View tab consists of a series of check boxes which, when checked, modify the initial state of the document window. These are not really

Boolean keys. If the key is present, the corresponding box in the UI will be checked, otherwise, the box remains cleared.

Key	Description
<code>fit</code>	Resize window to initial page
<code>center</code>	Center window on screen
<code>fullscreen</code>	Open in Full Screen mode
<code>showtitle</code>	Show document title in the title bar

Note that you can open the document in Full Screen mode using the `fullscreen` key above, or by using the `fullscreen` key of the `\setDefaultFS`. Either will work.

- `uioptions`: The User Interface Options region of the Initial View tab consists of a series of check boxes which, when, checked hide an UI control. These are not really Boolean keys. If the key is present, the corresponding box in the UI will be checked, otherwise, the box remains cleared.

Key	Description
<code>hidemenubar</code>	Hide menu bar
<code>hidetoolbar</code>	Hide tool bars
<code>hidewindowui</code>	Hide window controls

Important: The `hyperref` package can set some of these fields of the Initial View tab. The document author is *discouraged* from using `hyperref` to set any of these fields, though, usually they are overwritten by this package.

Example 43: We set the Initial View tab of the Document Properties dialog box.

```
\DeclareInitView
{%
  layoutmag={mag=ActualSize,navitab=UseOutlines,%
    openatpage=3,pagelayout=TwoPageLeft},
  windowoptions={fit,center,showtitle,fullscreen},
  uioptions={hidetoolbar,hidemenubar,hidewindowui}
}
```

`\DeclareInitView` is a companion command to `\DeclareDocInfo`. Each fills in a separate tab of the Document Properties dialog box. Use the package `aebxmp` to fill in advance metadata through `\DeclareDocInfo`.

12. Document Open Actions

You can set an action to be performed when the document is open, independently of the page the document is opened at.

```
\additionalOpenAction{<action>}
```

Command Description: The `<action>` can be any type of action described in the *PDF Reference*, but it is usually a JavaScript action.

Command Location: Place this command in the preamble.

The following example gets the time the user first opens the document

```
\additionalOpenAction{\JS{%
  var timestamp = util.printd("mm-dd-yy, H:MM:ss.", new Date());}}
```


Section 12: Document Open Actions

Important: This open action takes place rather early in document initialization, before the document level JavaScript is scanned; therefore, the *<action>* should not reference any document level JavaScript, as at the time of the action, they are still undefined. You are restricted to core JavaScript and the JavaScript API for **Acrobat**.

Using layers put a natural restriction on the version that can be used to effectively view the document. To put a requirement on the viewer to be used, use the `\requiresVersion` command.

```
\requiresVersion{<version_number>}
\requiredVersionMsg{<message>}
\alternateDocumentURL{<url>}
\requiredVersionMsgRedirect{<message>}
\afterRequirementPassedJS{<JS code>}
```

Command Location: Place these commands in the preamble.

Command Description: For the `\requiresVersion` command, the parameter *<version_number>* is the minimal version number that this document is made for. If the version number of the viewer is less *<version_number>*, an alert box appears, and the document is silently closed, if outside a browser, or redirected, if inside a browser.

When the document is opened outside a web browser and the version number requirement is not met, the message contained in `\requiredVersionMsg` appears in an alert box. The default definition is

```
\requiredVersionMsg{%
    This document requires Adobe Reader or Acrobat,
    version \requiredVersionNumber\space or later.
    The document is now closing.}
```

The argument of `\requiresVersion` is contained in the macro `\requiredVersionNumber`, and this macro should be used in the message, as illustrated above.

When the document is opened in a browser and the version number requirement is not met the message contained in `\requiredVersionMsgRedirect` appears in an alert box. The default definition is

```
\requiredVersionMsgRedirect{%
    This document requires Adobe Reader or Acrobat},
    version \requiredVersionNumber\space or later.
    Redirecting browser to an alternate page.}
```

The browser is redirected to the URL specified in the argument of `\alternateDocumentURL`, the default definition of which is

```
\alternateDocumentURL{http://www.acrotex.net/}
```

The command `\requiresVersion` uses `\additionalOpenAction`; if you want to combine several actions, including an action for checking for the version number, use `\afterRequirementPassedJS`. For example,

```
\afterRequirementPassedJS
{%
    var timestamp = util.printd("mm-dd-yy, H:MM:ss.", new Date());
}
```

The above code will be executed if the version requirement is passed.

You can use `\afterRequirementPassedJS`, for example, to put deadline to view the document; that is, if the document is opened after a pre-selected date and time, the document should close down (or redirected to an alternate web page).

Important: When using APB, the minimum required version is 7. Thus,

```
\requiresVersion{7}
```

should be issued in the preamble of any APB document available publicly.

13. Commands for Creating Layers

Adobe provides two different methods for creating Optional Content Groups (OCG), or layers; these are a SimpleOC and a nested OC. The sole reference for OCG is the *pdfmark Reference Manual* [3]; see Appendix B, titled “Distilling Optional Content” for complete details.

A SimpleOC marks all content up to the *next* SimpleOC as being in the same OCG.⁹ A SimpleOC does not need a matching command to delimit the content as one SimpleOC is delimited by the next one. Insert `\ocOff` to make subsequent content non-optional. Also, an `\ocOff` should be placed between a SimpleOC and a nested OC, this would exclude the nested OC from being part of the OCG of the previous SimpleOC.

A nested OC requires a terminating command that delimits the content belonging to that group.¹⁰ Content in the same group can be made visible or hidden.

- Each layer has a *name* and an *initial state*. Layers with the same name are in the same Optional Content group, so they become visible and hidden in tandem.

Most of the user accessible commands discussed in this section have an option for setting the initial state, and some have an option for setting the name of the layer. Some commands, such as `\Bld`, already introduced, have an automatic naming scheme.

- APB maintains a counter, `ocCnt` that is used in the automatic naming of a layer.

13.1. Simple OC

The command `\Bld` is the simplest command for creating a layer, it is a SimpleOC. Its content is delimited by the *next* `\Bld`

- `\Bld`

```
\Bld[⟨KV-pairs⟩] ⟨text⟩
```

Key-Value Pairs: Beginning with APB 2.0, this command has optional arguments.

1. `print`: (APB 2.1) A choice key, the permissible values of `print` are `true`, `false`, `always`, `never`, and `whenevervisible`. The first two, `true` and `false`, are legacy values from APB 2.0, see below; the third and fourth, `always` and `never`, are equivalent to `true` and `false`, respectively. The fifth choice, `whenevervisible`, is new, in this case, the layer prints only if it is visible. The terms `always`, `never`, and `whenevervisible` roughly correspond to the terms used in the user interface.

`print`: (APB 2.0) A Boolean, which if `true`, the default value, this layer will be printed even if it is hidden. Set `print=false` if you do not want this layer to be printed while hidden. It will be printed while visible.

2. The optional argument of `\Bld` also obeys the key-value pairs of sub-page navigation, refer to [Table 1](#), page 10.
3. `noNode`: A Boolean, which if `true`, no sub-page navigation node is created for this layer. The default is `false`.

Command Description: The initial state of this layer is *off* and there is no access to the initial state through this command. The `\Bld` command gives a name of `pg\thepage.apb\#oc\theocCnt` to the OCG it creates. You’ll note that the name has the page number built into it, and the current value of `ocCnt`, a running counter.

⁹In APB, the low-level command for a SimpleOC is `\@SOC`.

¹⁰The low-level commands for nested OC are `\b@OC` and `\e@OC`, the latter delimits the OCG begun by the former.

- **\tBld**

```
\tBld[⟨KV-pairs⟩] ⟨text⟩
```

Command Description: This SimpleOC is used for creating toggle layers. The commands `\toggleColor` and `\toggleText` use `\tBld`.

Key-Value Pairs: Beginning with APB 2.0, this command has optional arguments.

1. `initState`: Set the initial state of the layer. Values of `true` and `on` are recognized to make the layer initially visible; and `false` and `off` for making it initially hidden. The default state is `initState=false`.
2. `print`: (APB 2.1) A choice key, the permissible values of `print` are `true`, `false`, `always`, `never`, and `wheneverVisible`. The first two, `true` and `false`, are legacy values from APB 2.0, see below; the third and fourth, `always` and `never`, are equivalent to `true` and `false`, respectively. The fifth choice, `wheneverVisible`, is new, in this case, the layer prints only if it is visible. The terms `always`, `never`, and `wheneverVisible` roughly correspond to the terms used in the user interface.

`print`: (APB 2.0) A Boolean, which if `true`, the default value, this layer will be printed even if it is hidden. Set `print=false` if you do not want this layer to be printed while hidden. It will be printed while visible.

3. `noNode`: A Boolean, which if `true`, no sub-page navigation node is created for this layer.

Changes in Syntax: In the first version of APB, the `\tBld` command had the syntax

```
\tBld[true|false] ⟨text⟩
```

Values of `true` and `false` have been replaced by `initState=true` and `initState=false`, respectively.

The OCG name for a layer created by the “toggle build”, `\tBld`, is

```
pg\thepage.apb-toggle\#oc\theocCnt-\theocSeq
```

Note that the name depends on the page, the `ocCnt` and the value of another count register, `ocSeq`. The base name `apb-toggle` is used by the controlling navigation JavaScript for identifying toggle layers. For a toggle layer, the previous toggle layer is turned off before the next layer is turned on. The value of `ocSeq` keeps a running count toggle layers in the sequence; this number is used by the JavaScript to navigate the toggle layers.

Example 44: As mentioned before, `\tBld` is used by other commands such as `\toggleColor` and `\toggleText` to create toggled layers. A simple example of usage would be

```
\setcounter{ocSeq}{0}  
Are you ready? \tBld One! \tBld Two! \tBld Three! \tBld Let's Go!!!
```

Obviously, `\tBld` should be used as a building block for other toggle layers you might want to define. `\tBld` is used in the commands `\toggleColor`, `\toggleColor`, `\dimTxt` and `\dimUpTxt`

- **\fBld**

Acrobat Forms provides a bit of a problem, they are not part of content (they are placed on top of the content), and so are not included in an OCG, yet, the desired effect is to make form fields, such as those created by `exerquiz`, visible or hidden in tandem with a layer. Layers can have associated with them a JavaScript action, and so the layers themselves can show and hide set of form fields. All we have to do is to create a scheme for communicating the field names and creating the JavaScript actions.

The next command was designed to be used for form fields.

```
\fBld[⟨KV-pairs⟩]{⟨baseName⟩} ⟨text⟩
```

Section 13: Commands for Creating Layers

This is a SimpleOC originally designed to work with form fields, hence the name `\fBld` for form build; however, the same features were incorporated into `\bBld`, a nested OCG command.

Command Location: Used within a `slide` environment.

Parameter Description: This command takes one optional and one required parameter.

1. The first optional parameter takes any of several key-value pairs.

- **fields:** A comma delimited list of form fields to be associated with this layer. These fields will be made hidden initially, and become visible when the layer becomes visible; the fields become hidden when the layer becomes hidden. The names of the fields should be in double quotes. Here are a couple of examples:

```
\fBld[fields="myField1"]{q1}
\fBld[fields={"myField2","myField3","myField4"}]{q2}
```

If there is more than one field listed, then the entire list of fields needs to be enclosed in matching braces to avoid parsing error of `xkeyval`, as illustrated above.

- **sound:** The value of the sound key is a sound clip, already embedded in the document by `\importSounds`. This sound clip will play when the layer is made visible and not play when the layer is hidden.

```
\fBld[fields="myField1",sound=trek.wav]{q1}
```

Here, we associate a sound wave with the layer named `q1`. When this layer is made visible, the form field "myField1" will be made visible and the sound clip will play (for dramatic effect).

2. **baseName:** The unique (for this page) name for this layer. The actual (internal) name of the layer is `pg\thepage.apb<baseName>\#oc\theocCnt`. This command writes its name to the auxiliary file, this file is used by APB to process the `fields` and `sound` keys, and with `\setDisplayOrder`, which is discussed below, in 'Nested OCG' on page 61.

3. **print:** (APB 2.1) A choice key, the permissible values of `print` are `true`, `false`, `always`, `never`, and `whenever`. The first two, `true` and `false`, are legacy values from APB 2.0, see below; the third and fourth, `always` and `never`, are equivalent to `true` and `false`, respectively. The fifth choice, `whenever`, is new, in this case, the layer prints only if it is visible. The terms `always`, `never`, and `whenever` roughly correspond to the terms used in the user interface.

`print:` (APB 2.0) A Boolean, which if `true`, the default value, this layer will be printed even if it is hidden. Set `print=false` if you do not want this layer to be printed while hidden. It will be printed while visible.

4. (APB 2.0) Obeys the key-value pairs of sub-page navigation. See [Table 1](#), page 10.

5. **noNode:** (APB 2.0) A Boolean, which if `true`, no sub-page navigation node is created for this layer. The default is `false`.

Example 45: Below is a simple example of the use of the `fields` and `sound`.

```
\fBld[fields="myField1",sound=thunder.wav]{dps}
This is a form: \textField{myField1}{1in}{12bp}
```

- The `\fBld` has another interesting and useful feature: it obeys the `\setDisplayOrder` command, used for changing the order of display of a layer. Normally, the order of layers is the same order that `TEX` processes the layer building commands.

- `\ocOff`

```
\ocOff
```

Executing `\ocOff` in the text makes subsequent content non-optional. This command is executed in the running footer so that layers do not extend across page boundaries. You should place an `\ocOff` between the last SimpleOC and the next nested OC.

- The APB maintains a switch, `\ifapb@SimpleOCOn`, which is set to `true` when a SimpleOC begins. At the beginning of a nested OC, a check is made of this switch, if the switch is still set to `true`, a warning message is placed in the log and an `\ocOff` is executed, which terminates the scope of the last SimpleOC, and sets the switch to `false`. The document author should correct the problem with overlapping scope in the source file.

Note that `\ocOff` is different from `\turnOcOff`, described later on page 71; the former terminates the last SimpleOC group and makes subsequent text non-optional, the latter causes all OC code to be removed from all the OC commands.

13.2. Nested OCG

A nested OCG requires a terminating delimiter. Access to this delimiter is through the `\eBld` command.

- `\bBld`

The first nested OCG command to be considered is `\bBld`. I forgot what the ‘b’ stands for in the `\bBld` command, please forgive me. This command pair is identical in functionality to the SimpleOC, `\fBld`. Probably don’t need both. But, here it is.

```
\bBld[⟨KV-pairs⟩]{⟨baseName⟩} ⟨text⟩ \eBld
```

Command Location: Used within a `slide` environment.

Parameter Description: This command takes one optional and one required parameter.

1. The first optional parameter takes any of several key-value pairs.

- **fields:** A comma delimited list of form fields to be associated with this layer. These fields will be made hidden initially, and become visible when the layer becomes visible; the fields become hidden when the layer becomes hidden. The names of the fields should be in double quotes. Here are a couple of examples:

```
\bBld[fields="myField1"]{q1}  
\bBld[fields={"myField2","myField3","myField4"}]{q2}
```

If there is more than one field listed, then the entire list of fields needs to be enclosed in matching braces to avoid parsing error of `xkeyval`, as illustrated above.

- **sound:** The value of the `sound` key is a sound clip, already embedded in the document by `\importSounds`. This sound clip will play when the layer is made visible and not play when the layer is hidden.

```
\bBld[fields="myField1",sound=trek.wav]{q1}
```

Here, we associate a sound wave with the layer named `q1`. When this layer is made visible, the form field "myField1" will be made visible and the sound clip will play (for dramatic effect).

2. `baseName`: The unique (for this page) name for this layer. The actual (internal) name of the layer is `pg\thepage.apb⟨baseName⟩\#oc\thecCnt`. This command writes its name to the auxiliary file, this file is used by APB to process the `fields` and `sound` keys, and with `\setDisplayOrder`, which is discussed next.

Section 13: Commands for Creating Layers

3. `print`: (APB 2.1) A choice key, the permissible values of `print` are `true`, `false`, `always`, `never`, and `whenever`. The first two, `true` and `false`, are legacy values from APB 2.0, see below; the third and fourth, `always` and `never`, are equivalent to `true` and `false`, respectively. The fifth choice, `whenever`, is new, in this case, the layer prints only if it is visible. The terms `always`, `never`, and `whenever` roughly correspond to the terms used in the user interface.

`print`: (APB 2.0) A Boolean, which if `true`, the default value, this layer will be printed even if it is hidden. Set `print=false` if you do not want this layer to be printed while hidden. It will be printed while visible.

4. (APB 2.0) Obeys the key-value pairs of sub-page navigation. See [Table 1](#), page 10.
5. `noNode`: (APB 2.0) A Boolean, which if `true`, no sub-page navigation node is created for this layer. The default is `false`.

```
\setDisplayOrder{<baseName_1>[<KV-pairs_1>],<baseName_2>[<KV-pairs_2>],...,  
  <baseName_n>[<KV-pairs_n>]}
```

Command Description: Changes the order the layers appear, in the order listed as the arguments of this command. The `\setDisplayOrder` command can be used to set a different order for the build commands `\fBld` and `\bBld`.

Key-Value Pairs: APB 2.0 extends the arguments of `\setDisplayOrder` to include optional key-value pairs for sub-page navigation. The key-value pairs of sub-page navigation. See [Table 1](#), page 10. If you specify several key-value pairs, to avoid parsing errors by the `xkeyval` package, you should enclose the who parameter in braces, like so,

```
\setDisplayOrder{{layer3[nexttrans=FlyIn,prevtrans=Fade]},layer2,layer1}
```

Note the braces around `layer3` and its optional arguments, this is necessary because the optional arguments contain a comma.

Command Location: Used in any `slide` environment, and *should be placed before* any of the `\fBld` and `\bBld` commands it references in its arguments.

Parameter Description: The command takes one argument, a comma delimited list (with no spaces between the comma and the `baseName`). Each of the base names should correspond to a name given to a `\fBld` or `\bBld` command on the current page. The order of the names listed in the argument is the order the layers with those names appear on the page.

Example 46: The following is a simple example.

```
\setDisplayOrder{layer3,layer2,layer1}  
...  
\bBld{layer1} This line will appear following the appearance  
of layer3 and layer2\eBld  
...  
\fBld{layer2} This line will appear following the appearance  
of layer3\ocOff  
...  
\bBld{layer3} This will be the first to appear of the  
three layers!\eBld  
...
```

There can be other `\Bld`, `\fBld`, `\bBld`, etc., commands between these three, of course.

Example 47: When the `spnavi` option is taken, you need to avoid creating more than one sub-page navigation node per layer. So, the above example should read:

Section 13: Commands for Creating Layers

```
\setDisplayOrder{layer3[nextttrans=FlyIn],layer2,layer1}
...
\bBld[noNode]{layer1} This line will appear following the appearance
of layer3 and layer2\eBld
...
\fBld[noNode]{layer2} This line will appear following the appearance
of layer3\ocOff
...
\bBld[noNode]{layer3} This will be the first to appear of the
three layers!\eBld
...
```

The next command, `\refBld` applies to optional content created by the `\fBld` and `\bBld` commands. This command, a nested OC, allows you to place new content into an already defined group.

```
\refBld{<baseName>} <text> \eBld
```

Command Location: Used in any `slide` environment.

Parameter Description: Here, `<baseName>` is the name of a layer assigned by `\fBld` or `\bBld`.

Example 48: The example below first displays the layer with the base name of `apb`, the content consisting of ‘The AcroTeX Presentation’ and ‘Bundle’, then it shows the word ‘The Great!’ between them.

```
\begin{slide}[fontseries=bfseries,fontsize=large]
\begin{center}
\bBld{apb}\textcolor{blue}{The AcroTeX Presentation}\eBld\[\[2ex]
\Bld\textcolor{red}{The Great!}\ocOff\[\[2ex]
\refBld{apb}\textcolor{blue}{Bundle}\eBld
\end{center}
\end{slide}
```

There may be the occasion you want to set the JavaScript action for `\bBld` (or `\fBld`). The command `\trueName` can be used to access the true name of the layer.

```
\trueName(<baseName>)
```

Notice the matching parentheses around `<baseName>`, such as `\trueName(myLayer)`.

This command can be used in the `insDLJS`, `addJSToPageOpen` or `addJSToPageClose` environments to reference the layer name of the layer. (In these environments the catcode of the left and right braces have been changed, so the usual method of enclosing an argument in braces does not work.)

Example 49: Below is a simple example of usage, when the layer with base name of `alert` is made visible, an alert box appears with the message ‘This is Planet AcroTeX!’. Here, we use the `insDLJS` environment in the preamble to write our JavaScript and assign it to the desired layer. The `assocLayers` array takes the name of a layer and returns the index in the `ocgs` array. The `ocgs` array contains all the OCG objects (with some exceptions) contained in the document. We get the OCG object of our layer and assign the JS action with the `setAction` method of the OCG object. See the *Acrobat JavaScript Scripting Reference* [1] for details of the OCG object.

```

\begin{insDLJS}[_setActions]{setActions}{setActions}
var _setActions = true;
var i = assocLayers["\trueName(alert)"];
ocgs[i].setAction("if (ocgs["+i+"].state)app.alert('This is Planet AcroTeX!')");
\end{insDLJS}

\begin{document}
\begin{slide}
\Bld This is an example \ocOff \bBld{alert}of using a OCG action
to execute JavaScript. \eBld\Bld Will it work?\ocOff
\end{slide}

```

- **\rBld**

The next command is for creating “raw” builds, anything goes, very little control is imposed on this command by APB.

```
\rBld[⟨KV-pairs⟩]{⟨baseName⟩} ⟨text⟩ \eBld
```

Command Location: Used in any `slide` environment.

Key-Value Pairs: Beginning with APB 2.0, this command recognizes a number of new key-value pairs.

1. `initState`: Set the initial state of the layer. Values of `true` and `on` are recognized to make the layer initially visible; and `false` and `off` for making it initially hidden. The default state is `initState=false`.
2. `print`: (APB 2.1) A choice key, the permissible values of `print` are `true`, `false`, `always`, `never`, and `whenever`. The first two, `true` and `false`, are legacy values from APB 2.0, see below; the third and fourth, `always` and `never`, are equivalent to `true` and `false`, respectively. The fifth choice, `whenever`, is new, in this case, the layer prints only if it is visible. The terms `always`, `never`, and `whenever` roughly correspond to the terms used in the user interface.

`print`: (APB 2.0) A Boolean, which if `true`, the default value, this layer will be printed even if it is hidden. Set `print=false` if you do not want this layer to be printed while hidden. It will be printed while visible.
3. (APB 2.0) Obeys the key-value pairs of sub-page navigation. See [Table 1](#), page 10.
4. `noNode`: (APB 2.0) A Boolean, which if `true`, no sub-page navigation node is created for this layer. The default is `false`.

Parameter Description: The second required parameter, `⟨baseName⟩`, is the base name for the layer, its actual name is `pg\thepage.⟨baseName⟩`. The inclusion of the page number restricts the content of the layer to the current page.

Changes in Syntax: In the first version of APB, the `\rBld` command had the syntax

```
\rBld[true|false]{⟨baseName⟩} ⟨text⟩ \eBld
```

Values of `true` and `false` have been replaced by `initState=true` and `initState=false`, respectively.

- The primary advantages of this raw build is that you can easily create content that is in the same group and can easily rearrange the order of display, without writing to the auxiliary file as `\fBld` and `\bBld` do. Layers with the same name (and in the same slide) are in the same group.

Example 50: Here is a simple example to illustrate. Note the first line, we layout two empty layers in the order we want them to appear, then later, we add to each content group. (Tricky, huh! This is how `\setDisplayOrder` works, by the way.)

Section 13: Commands for Creating Layers

```
\rBld{h2}\eBld\rBld{h1}\eBld
```

Let us change \rBld{h1}the order of \eBld of display for \rBld{h2}this sentence.\eBld

First we display \rBld{h2}the \texttt{h2} layer\eBld, then we display \rBld{h1}the \texttt{h1} layer\eBld.

The same effect can be obtained from \bBld, but it has the overhead of writing to the .aux file.

Example 51: If the `spnavi` option is taken, the above example needs to be modified so that duplicate sub-page navigation nodes will not be created.

```
\rBld{h2}\eBld\rBld{h1}\eBld
```

Let us change \rBld[noNode]{h1}the order of \eBld of display for \rBld[noNode]{h2}this sentence.\eBld

First we display \rBld[noNode]{h2}the \texttt{h2} layer\eBld, then we display \rBld[noNode]{h1}the \texttt{h1} layer\eBld.

• \xBld

The command \xBld is different from \Bld, \tBld, \fBld, \bBld and \rBld. These are all designed to create “talking points” or special effects, all subject to control by the navigation panel. The layers created by \xBld are invisible to the navigation buttons and does not spawn any sub-page navigation nodes, and are meant to be controlled by hypertext links from within the body of the text.

The ‘x’ stands for cross-reference, or cross-pages. With \xBld you can set a content group over several pages, make it visible or hidden using links you create.

```
\xBld[⟨KV-pairs⟩]{⟨baseName⟩} ⟨text⟩ \eBld
```

Key-Value Pairs: Beginning with APB 2.0, this command recognizes two new key-value pairs.

1. `initState`: Set the initial state of the layer. Values of `true` and `on` are recognized to make the layer initially visible; and `false` and `off` for making it initially hidden. The default state is `initState=false`.
2. `print`: (APB 2.1) A choice key, the permissible values of `print` are `true`, `false`, `always`, `never`, and `whenevervisible`. The first two, `true` and `false`, are legacy values from APB 2.0, see below; the third and fourth, `always` and `never`, are equivalent to `true` and `false`, respectively. The fifth choice, `whenevervisible`, is new, in this case, the layer prints only if it is visible. The terms `always`, `never`, and `whenevervisible` roughly correspond to the terms used in the user interface.

`print`: (APB 2.0) A Boolean, which if `true`, the default value, this layer will be printed even if it is hidden. Set `print=false` if you do not want this layer to be printed while hidden. It will be printed while visible.

Parameter Description: The second required parameter, `⟨baseName⟩`, is the base name for the layer, its actual name is `xb.⟨baseName⟩`. Notice that there is no page number in the name, this makes it easy to create optional content groups extending over multiple pages. Layers with the same name will be in the same group; different names will be in different groups.

Changes in Syntax: In the first version of APB, the \xBld command had the syntax

```
\xBld[true|false]{⟨baseName⟩} ⟨text⟩ \eBld
```

Section 13: Commands for Creating Layers

Values of `true` and `false` have been replaced by `initState=true` and `initState=false`, respectively.

Command Location: Used in any `slide` environment.

 The file `apb_xBld.tex` demonstrates the use of `\xBld`, with suggested applications.

- **`\animeBld`**

We now come to one of the more exciting features of the APB, animation. Here, animation means creating a sequence of layers (frames) then providing the control over the layers to sequentially show and hide each frame at regular time intervals.

This difference between this scheme and traditional methods is that APB creates one page with 50 frames (say), while other packages would create 50 pages. Two possible advantages: (1) A single page with 50 frames is smaller than 50 pages (there is a large overhead in the creation of each page); it is my belief that frames can be flipped faster than pages can be turned. These claims, however, are not substantiated, but reasonable.

The `\animeBld` is actually a `\rBld` with a particular name, a name that is recognized by the background JavaScript. The navigation routines, such as 'Next' and 'Prev' skip over these special animation layers. The animation layers themselves are controlled by their own set of buttons.

```
\animeBld[⟨KV-pairs⟩] ⟨text⟩ \eBld
```

Key-Value Pairs: Beginning with APB 2.0, this command recognizes two key-value pairs `noNode` key.

1. `initState`: Set the initial state of the layer. Values of `true` and `on` are recognized to make the layer initially visible; and `false` and `off` for making it initially hidden. The default state is `initState=false`.
2. `print`: (APB 2.1) A choice key, the permissible values of `print` are `true`, `false`, `always`, `never`, and `whenever`. The first two, `true` and `false`, are legacy values from APB 2.0, see below; the third and fourth, `always` and `never`, are equivalent to `true` and `false`, respectively. The fifth choice, `whenever`, is new, in this case, the layer prints only if it is visible. The terms `always`, `never`, and `whenever` roughly correspond to the terms used in the user interface.

`print`: (APB 2.0) A Boolean, which if `true`, the default value, this layer will be printed even if it is hidden. Set `print=false` if you do not want this layer to be printed while hidden. It will be printed while visible.

Because an animation is meant to be controlled by a button, or activated by a JavaScript function, `\animeBld` does not create any sub-page navigation nodes.

Command Location: Used in any `slide` environment.

Parameter Description: The optional parameter takes an argument of `true` or `false`, with a default of `false`. If set to `true`, the layer is initially visible. The `⟨text⟩` is the content of your frame, perhaps one of a whole sequence of frames.

Before creating your animation, you need to declare the animation

Changes in Syntax: In the first version of APB, the `\rBld` command had the syntax

```
\animeBld[true|false] ⟨text⟩ \eBld
```

Values of `true` and `false` have been replaced by `initState=true` and `initState=false`, respectively.

```
\DeclareAnime{⟨baseName⟩}{⟨time⟩}{⟨nFrames⟩}
```

Command Location: Used in any `slide` environment.

Parameter Description: Where `baseName` is the name common to all the frame layers, `nFrames` is the number of frames in the animation, and `time` is the length in milliseconds that each frame should be shown before it is hidden and next one in the sequence is shown.

`\DeclareAnime` does nothing other than to define four macros that are used by the control buttons. Three of these are `\animBaseName`, `\animSpeed`, `\nFrames`, the values of which hold the values of the first, second and third parameters; the last macro sets the name of the animation:

```
\newcommand{\animeBldName}
  {apb-special.\animBaseName\#\theocSeq-\nFrames}
```

It is our experience that PSTricks (by Timothy Van Zandt, Denis Girou & Herbert Voß), along with all the various contributed packages, is the ideal tool for drawing sophisticated animations. Another very useful package is the `fp` package (by Michael Mehlich *et al.*) for performing real arithmetic.

Animation is a very attractive concept and a useful platform for illustrating mathematical or physical concepts. It is hoped that many of our users, if any, explore anime, and even develop new techniques and identify additional packages for creating wonderful and awe-inspiring anime.

A super energetic and inspired member of our ASDT ([AcroTeX Software Development Team](#)) has created numerous animations, some of them quite amazing. These are in the `examples/anime` folder. We present here only the simplest and shortest of animations.

ex13 Example 52: This anime takes the phrase “**AcroTeX Presentation Bundle!**” and moves it horizontally across the page. There are two control buttons, one for shifting left, the other for shifting right.

```
\begin{slide}
  \begin{large}\color{red}
  \def\mytxt{\AcroTeX Presentation Bundle!}
  \DeclareAnimation{shift}{10}{40}
  \noindent\multido{\dShift=0bp+2bp}{40}{%
    \animeBld\makebox[0pt][l]{\kern\dShift\mytxt}\eBld}
  \end{large}
  \begin{center}
    \aniGraphicsButtonB\aniGraphicsButtonF
  \end{center}
\end{slide}
```

The example uses `\multido` (from the `multido` package, part of PSTricks) to create a series of frames using `\animeBld`; each frame has the phrase moved over a little bit more.

There are two methods of controlling an animation, by using form buttons or graphics buttons. The above anime uses graphics buttons (`\aniGraphicsButtonB` and `\aniGraphicsButtonF`). The list of form buttons are listed below. There are a total of ten buttons that can be used to control animation.

```
\aniFormsButtonF[⟨appearance⟩]{⟨width⟩}{⟨height⟩}
\aniFormsButtonB[⟨appearance⟩]{⟨width⟩}{⟨height⟩}
\aniFormsButtonC[⟨appearance⟩]{⟨width⟩}{⟨height⟩}
\aniFormsButtonP[⟨appearance⟩]{⟨width⟩}{⟨height⟩}
\aniFormsButtonSF[⟨appearance⟩]{⟨width⟩}{⟨height⟩}
\aniFormsButtonSB[⟨appearance⟩]{⟨width⟩}{⟨height⟩}
\aniFormsButtonPlay[⟨appearance⟩]{⟨width⟩}{⟨height⟩}
\aniFormsButtonPlus[⟨appearance⟩]{⟨width⟩}{⟨height⟩}
\aniFormsButtonMinus[⟨appearance⟩]{⟨width⟩}{⟨height⟩}
\aniFormsButtonResetDelay[⟨appearance⟩]{⟨width⟩}{⟨height⟩}
\displayAnimeSpeed
```

Section 13: Commands for Creating Layers

Parameter Description: Each of these commands takes four parameters

1. **appearance:** This optional parameter is used to change the appearance of the button, see the [eforms](#) documentation for details.
2. **width:** The width of the button to be created. If left empty, the eforms package attempts to compute the width for you based on the value of the \CA key, see [Example 52](#), page 67.
3. **height:** The height of the button to be created.

These descriptions of these buttons are the same as their graphical counterparts, to be described next. Some standard combinations of buttons are bundled together for ease of use.

```
\aniFormsConsoleBasic  
\aniFormsConsoleI  
\aniFormsConsoleII
```

The descriptions of these animation controls are again deferred. There are graphical counterparts with the same functionality.

There is a set of graphical buttons that are available as well. These are simpler to use and are consistent with the design of many of the other graphics that come with APB. These are...

```
\aniGraphicsButtonF  
\aniGraphicsButtonB  
\aniGraphicsButtonC  
\aniGraphicsButtonP  
\aniGraphicsButtonSF  
\aniGraphicsButtonSB  
\aniGraphicsButtonPlay  
\aniGraphicsButtonPlus  
\aniGraphicsButtonMinus  
\aniGraphicsButtonResetDelay  
\displayAnimeSpeed
```

These commands have no arguments, they use the macros created by \DeclareAnime.

- \aniGraphicsButtonF: Starts the anime forward, from the first frame to the last.
- \aniGraphicsButtonB: Starts the anime backward, from the last frame to the first.
- \aniGraphicsButtonC: Clears the anime back to its initial state.
- \aniGraphicsButtonP: Pauses an anime, preserving its current state. The animation will continue when the forward, backward or play button is pressed.
- \aniGraphicsButtonSF: Steps forward through the anime one frame at a time.
- \aniGraphicsButtonSB: Steps backward through the anime one frame at a time.
- \aniGraphicsButtonPlay: Plays an anime. If the anime is stopped, will play forward. If the anime is paused, it will continue the anime in the direction before the pause.
- \aniGraphicsButtonPlus: Increases the time delay of the anime. Pressing this button will cause the anime to be played at a *slower* speed.
- \aniGraphicsButtonMinus: Decreases the time delay of the anime. Pressing this button will cause the anime to be played at a *faster* speed.
- \aniGraphicsButtonResetDelay: Clicking on this button causes the speed control to return to its default setting.

Section 13: Commands for Creating Layers

- `\displayAnimeSpeed`: This is a text field that shows the current time delay in milliseconds.

These various buttons, which are available individually, have been bundled together in a more or less “natural” grouping.

```
\aniGraphicsConsoleBasic
\aniGraphicsConsoleI
\aniGraphicsConsoleII
```

Description:

- `\aniGraphicsConsoleBasic` is the basic anime console consisting of the backward, clear and forward buttons, in that order (`\aniGraphicsButton<ext>` with extensions B,C, and F).
- `\aniGraphicsConsoleI` consists, from left to right, of `\aniGraphicsButton<ext>` with the extensions of B, SB, C, SF, F, in that order. Finally,
- `\aniGraphicsConsoleII` is comprised of the `\aniGraphicsButton<ext>` buttons with extensions Minus, Play, ResetDelay, P, Plus. Slightly to the right of the Plus button is the `\displayAnimeSpeed` text field that displays the current time delay.

ex13 Example 53: This anime draws the graph of the sine function over $[0, 2\pi]$. In the preamble we have used packages `pstricks-add` and `fp`.

```
\begin{slide}
\DeclareAnimation{graph}{10}{40}
\newcommand*\animeFrame{\animeBld\psplot[linecolor=red]{0}{\xi}{\sin(x)}\eBld}
\psset{llx=-12pt, lly=-12pt,urx =12pt,ury =12pt,xAxisLabel=$x$,yAxisLabel=$y$}
\begin{psgraph*}[arrows=->,trigLabels=true,trigLabelBase=2,
  dx=\psPiH](0,0)(-.5,-1.5)(6.75,1.5){.5\linewidth}{4cm}
  \psset{algebraic=true}
  \rput(4,1){$y=\sin(x)$}
  \FPdiv{\myDelta}{\psPiTwo}{\nFrames}\renewcommand*\xi{0}%
  \multido{\i=1+1}{40}{\FPadd{\xi}{\xi}{\myDelta}\animeFrame}
\end{psgraph*}

  \aniGraphicsConsoleBasic
\end{slide}
```

Note that we have declared a total of 40 frames, yet we divide the interval 6.2832 by 39. The first from is the graph from 0 to ξ (set initially to zero), we then define $\xi = \xi + \text{myDelta}$, and graph from 0 to ξ for 39 times, each time updating the value of ξ . The reason we include the graph from 0 to 0 is so that when the reverse button is pressed, the sine will “un-graph” itself to no visible curve.

There is a selection of two colors for the anime graphics buttons, gray (the default) and black.

```
\useGrayButtons
\useBlackButtons
```

Command Location: Anywhere in the document to set the color of the next set of anime graphics buttons.

- `\stBld`

One of the options for APB is `sideshow`, as opposed to slide show. A sideshow is a series of layers—perhaps unrelated to the content of the document—that appear as the viewer works through the presentation. The motivating application for the sideshow is the creation of a *thermometer*, a graphical

gauge that indicates the progress through the document. The `\stBld` command is used to create a sideshow

(APB 2.0) The implementation of this feature has changed for version 2.0. Side show is now only available when you use the `snavi` option and while in *fullscreen* viewing mode. The side show feature is now controlled by sub-page navigation actions. The side show is much more robust, and responds more quickly than the old method of using JavaScript. The side show advances only for build commands that generate a sub-page navigation node and have a name ending with `#oc<number>`. This includes the structured commands (`\dBld`, `\dPt`, `\ddPt`, etc.) and common build commands (`\Bld`, `\bBld` and `\fBld`); remember, these builds have a `noNode` parameter that can cause APB not to create a sub-page navigation node. Typically, it excludes the special effects builds such as `\rBld`, `\animeBld`, `\xBld` and `\stBld`.


```
\stBld[true|false] <text> \eBld
\def\incEveryStBld{<number>}
```

Command Location: The use of this command should occur in the preamble of the document.

Parameter Description: The only parameter is optional. This parameter is a Boolean, `true` if the layer is to be visible initially, and `false` if the layer is to be hidden. The default is `false`. The name of the layer is automatically assigned, it is `st.apb-stBld\#\thestCnt`. Notice `\stBld` has its own counter, `stCnt`.

When the `sideshow` option is in effect, APB at the end of the document writes the final value of the counter `ocCnt` to the auxiliary file and the command `\nocCnt` contains the final count. This number `\nocCnt` is used in the construction of the side show.

The value of the `\incEveryStBld` macro can be used to control the number of layers created by `\stBld`. The default definition is `\def\incEveryStBld{1}`. This means you get an `\stBld` layer for each layer that has an `ocCnt`. In the preamble, `\def\incEveryStBld{2}`, declares that you want to get a `\stBld` for every second layer with an `ocCnt`. However, nothing is automatic, this must be programmed into the code.


 **Example 54:** The following example was taken from the file `apb_stBld.tex`. In this file, two thermometers are built.

In the code of [Figure 7](#), we create a thermometer that appears in the logo portion of the navigation panel. The thermometer is a circular disk filled by a colored wedge whose area is proportional to the proportion of the total document the user has viewed. The thermometer does not count pages, it counts all the build commands that are counted by `ocCnt` (this would exclude animation layers, for example).

Comments: (7) On page one, we have a logo appear. (8) Set the `stCnt` to 0, and set the desired units for the `pspicture` environment of `PSTricks`. (10) Compute

$$\backslash nCnt = \backslash noocCnt / \backslash incEveryStBld$$

which we round off to an integer. This is the total number of `\stBld` layers we want to create. (11) Compute the incremental degrees `\incDeg = 360 / \nCnt` for the central angle of our circular disk. (12) Set up our `pspicture` environment. (13) Draw a circular disk. (14) Now begin a `\multido` to create `\nCnt` layers using `\stBld` filling up the disk with wedges. The wedge goes from 0 to `\apbDeg`, which is computed to be `\apbDeg = \i \cdot \incDeg`, where `\i` is the counter of the `\multido`. (16) Close the `pspicture` environment. Finally, (20) we take the `\diskThermometer` command just defined, and use it as the logo using `\insertLogo`.

 There is a Boolean switch `\ifsideshow` that can be used to do conditional work. Have it one way if for a side show, and another if not.

```

1 ...
2 \usepackage{apb}
3 \usepackage{pstricks-add}
4 \usepackage{fp}
5 ...
6 \newcommand{\diskThermometer}{%
7 \ifnum\arabic{page}=1\relax\includegraphics[scale=.15]{APB_Logo}\else
8 \setcounter{stCnt}{0}\psset{unit=.5in}\@ifundefined{nocCnt}{}%
9   {%
10     \FPdiv{\nCnt}{\nocCnt}{\incEveryStBld}\FPround{\nCnt}{\nCnt}{0}%
11     \FPdiv{\incDeg}{360}{\nCnt}\FPclip{\incDeg}{\incDeg}\centering%
12     \noindent\begin{pspicture}(-.5,-.5)(.5,.5)%
13     \pswedge[fillstyle=solid,fillcolor=webye1low,linewidth=0pt](0,0){.5}{0}{360}%
14     \multido{\i=1+1}{\nCnt}{\FPmul{\apbDeg}{\i}{\incDeg}\FPclip{\apbDeg}{\apbDeg}%
15     \stBld\pswedge[fillstyle=solid,fillcolor=red,linewidth=0pt](0,0){.5}{0}{\apbDeg}\eBld}%
16     \end{pspicture}%
17   }%
18 \fi}
19 \insertLogo{\diskThermometer}
20 ...
21 \begin{document}

```

Figure 7: Listing for Example 54

13.3. Turning OC On and Off

```
\turnOcOn \turnOcOff
```

APB defines an internal Boolean switch `\ifapb@ocOn`. All OC commands are surrounded by an `\if/\fi` construct. When the document author uses the `forpaper` option, for example, `\ifapb@ocOn` is set to false, which strips away all OC commands. The command pair `\turnOcOn` and `\turnOcOff` provides user access to turning this switch on or off.

If you place `\turnOcOff` in the *preamble*, this will globally turn OC off throughout the whole document, it can't be *turned on later*. This is because the OC start up code is not placed in the file at the `\begin{document}`. This may be useful if the document author wants to use the many features of APB without using layers.

If `\turnOcOff` is placed between slides, the effects are global. Any subsequent OC commands will not create any layers until `\turnOcOn` is encountered. `\turnOcOn` will also be global between slides.

You can say `\turnOcOff` between paragraphs, or for a whole slide (changes are local), in this way you can use the structured commands such as `\dBld`, and `\dPt` in a normal typesetting way. You can then insert `\turnOcOn` to restore OC support for the next OC command you use.

14. Some Special Effects

14.1. Simple Special Effects

- **Toggle and Dimming Color**

A special effect is to have a text fragment and to change its color for emphasis. Use the `\toggleColor` for this purpose.

```
\toggleColor*{<n_colors>}{<text>}{<color1>}...{<colorn>}
```

Parameter Description: The first parameter, '*' is optional; if present, the text is initially visible. The second parameter `<n_colors>` is the number of different colors that are to be used. Next comes the text, `<text>`, whose color is to be toggled, and finally comes the list of named colors. The `<text>` should not contain any verbatim text.

Command Location: Used in a `slide` environment.

ex8 **Example 55:** Example of usage:

```
\toggleColor*{3}{AcroTeX Presentation Bundle}{red}{green}{blue}
```

On a related subject, we can use the `\toggleColor` command to create a simple set of commands for dimming text.

```
\dimUpTxt*{<text>}{<color>}
\dimTxt*{<text>}{<color>}
```

Command Location: Used in a `slide` environment.

Parameter Description: The first parameter, ‘*’ is optional; if present, the text is initially visible. Next comes the text, `<text>`, whose color is to be dimmed, and finally comes `<color>`, the color we are to dim down from or up to. The `<text>` should not contain any verbatim text.

ex8 **Example 56:** Example of usage:

```
Let's \dimUpTxt*{dim up some text}{red} now
let's \dimTxt*{dim down some text}{blue}
```

☞ You can place `<text>` in a `\parbox` or in a `minipage` environment to dim an entire paragraph. Verbatim text causes problems and should be avoided. See `apb_ex8.tex` for an example of this.

• Toggling Text

Another special effect is to have a list of text fragments that can be toggled through. The widest text is put into an `\hbox` to determine the width to allocate for all the text. This material cannot, therefore, be wrapped around a line.

```
\toggleText*[[<KV-pairs>]]{<n_text>}{<text1>}...{<textn>}
```

Key-Value Pairs: (APB 2.0) This command now recognizes two key value pairs.

1. `widesttext`: The value of this key is a copy of the widest text that will be listed (this text is put into a `\hbox`); if this parameter is not present, then the `<text1>` is used to determine with width of the `\hbox`.
2. `noNode`: If this parameter is specified, no sub-page navigation node is created for any of the layers created by `\toggleText`. The `noNode` parameter is used for special effects, such as synchronizing several toggle fields, as described below.

Command Location: Used within a `slide` environment.

Parameter Description: The first parameter, ‘*’ is optional; if present, the text is initially visible. Following the optional parameters, if any, comes a specification, `<n_text>`, of the number of fragments that are to be used. Finally, comes the list of fragments; the number of these parameters equals `<n_text>`.

Changes in Syntax: In the first version of APB, the `\toggleText` command had the syntax

```
\toggleText[*][<widest_text>]{<n_text>}{<text1>}...{<textn>}
```

`<widest_text>` has been replaced by the key-value pair `widesttext=<widest_text>`, in addition to a new `noNode` parameter.

ex8 **Example 57:** Example of usage:

```
This is the \toggleText*[AcroTeX Presentation Bundle]{3}
{APB}{Planet AcroTeX}{AcroTeX Presentation Bundle}.
```

It is possible to set up several synchronized toggle fields. These fields will come on and go off together. All you have to do is to place the text in the same Optional Content Group (OCG). We have some special commands for doing this.

```
\saveTogTtxtOcCnt{<name>}
\useTogTtxtOcCnt{<name>}
\restoreTogTtxtOcCnt{<name>}
```

Command Location: Used in a `slide` environment.

Parameter Description: The only parameter is the $\langle name \rangle$. This name must be unique to the page. It will be used to group together toggle fields with the same name.

ex8 **Example 58:** In this example, found in `apb_ex8.tex`, links together two toggle fields created by the `\toggleText` command. The example illustrates how to use the three commands. First, use the command `\saveTogTtxtOcCnt` just before the first usage of `\toggleText` on the page. Now, when we create another `\toggleText`, we first place a `\useTogTtxtOcCnt` before the `\toggleText`, and finish up with `\restoreTogTtxtOcCnt`. Repeat as desired. (APB 2.0) If this example is compiled using the `snavi` option, then to make it work correctly, we don't want to duplicate nodes of layers already created, so we use the `noNode` option for the second set of toggle fields.

```
\saveTogTtxtOcCnt{dps}          % save the current ocCnt
\toggleText[\parbox{\linewidth}{\hfill}]{3}
{\parbox{\linewidth}{\begin{equation}x + y = 1\label{eq1}\end{equation}}}{
{\parbox{\linewidth}{\begin{equation}x^2 + y^2 = 1\label{eq2}\end{equation}}}{
{\parbox{\linewidth}{\begin{equation}x^3 + y^3 = 1\label{eq3}\end{equation}}}}

\Bld This is some text

\useTogTtxtOcCnt{dps}          % Use previously saved ocCnt
\toggleText[noNode]{3}        % noNode needed when using snavi option
{Equation~\eqref{eq1} is a linear equation.}
{Equation~\eqref{eq2} is a quadratic equation.}
{Equation~\eqref{eq3} is a cubic equation.}
\restoreTogTtxtOcCnt{dps}      % and restore

\Bld Some more text
```

Several of these “linked” toggle fields can appear on a page.

- For larger displays of graphics and text that can be toggled through, use `\toggleGraphics` and `\toggleCaptions`, designed for coordinated toggling.

14.2. Creating a Graphics/Captions Slide Show

On certain news web sites it is not unusual to see a slide show of the days news photos with accompanying captions. This inspired me to develop a PDF version of these displays. It was for this purpose that `\toggleGraphics` and `\toggleCaptions` were developed.

There is not much of a difference between these two commands, except the first is designed to display a sequence of graphics in a viewing window, whereas the other is meant to be used to display text, perhaps describing the photos. These can be used singly or in tandem.

```
\toggleGraphics[⟨KV-pairs⟩]{⟨width⟩}{⟨height⟩}{⟨baseName⟩}{⟨nPics⟩}  
  {⟨graphic1⟩}...{⟨graphicn⟩}
```

Command Location: Used with a `slide` environment.

Parameter Description: `\toggleGraphics` takes five parameters, only the first is optional.

1. **key-values:** This optional parameters several (optional) key-value pairs.
 - **poster:** A Boolean, which if present (or is set equal to `true`), will cause the first graphic to be displayed initially.
 - **navibar:** A Boolean, which if present, will enable to toggling of the graphics by the navigation buttons; if `false`, or not present, the controls are expected to be separate buttons. Note: This key must appear in every toggle created by `\toggleGraphics` and `\toggleCaptions` with the same `baseName`.
 - **halign:** Horizontal alignment within the viewing window. Permissible values are `l`, `c` and `r`, for left, center and right alignment, respectively. The default value is `c`, center.
 - **valign:** Vertical alignment within the viewing window. Permissible values are `t`, `c` and `b`, for top, center and bottom alignment, respectively. The default value is `t`, top.
2. **width:** The width of the viewing window; this value is a length.
3. **height:** The height of the viewing window; this value is a length.
4. **baseName:** A unique (to the page) name for the toggle fields; this name is used to build the name of the layers.
5. **nPics:** The number of graphics (or pictures) that are to be displayed in this toggle field.
6. **graphic₁...graphic_n:** Following the declaration of the number of graphics (`nPics`) comes a listing of the graphics, there should be `nPics` of them. Each of these `nPics` are enclosed in their own set of matching braces.

```
\toggleCaptions[⟨KV-pairs⟩]{⟨width⟩}[⟨height⟩]{⟨baseName⟩}{⟨nCaps⟩}  
  {⟨text1⟩}...{⟨textn⟩}
```

Command Location: Used with a `slide` environment.

Parameter Description: `\toggleCaptions` takes five parameters, the first and third are optional.

1. **key-values:** Same key-values as `\toggleGraphics`.
2. **width:** The width of the viewing window; this value is a length.
3. **height:** The (optional) height of the viewing window; this value is a length. If specified contents are placed in a `\parbox` with depth specification (second optional parameter of `\parbox`); if the optional parameter has a value of `\natheight`, contents are placed in a `\parbox` with no depth specification; finally, if this parameter is not specified, contents are placed in a `\makebox`;
4. **baseName:** A unique (to the page) name for the toggle fields; this name is used to build the name of the layers.
5. **nCaps:** The number of captions (or simple text) that are to be displayed in this toggle field. If `\toggleCaptions` has the same `baseName` as a `\toggleGraphics`, the values of `nPics` should be the same for the two.

6. `text_1...text_n`: Following the declaration of the number of captions (`nPics`) comes a listing of the captions, there should be `nPics` of them. Each of these `nPics` are enclosed in their own set of matching braces. The contents of each argument can be text or even other graphics. Content will be typeset in a `\parbox`.

ex9 Example 59: The tutorial file `apb_ex9.tex` demonstrates these two commands. The first section of that document uses the `navibar` option. In this case, toggling is done using the ‘Next’ button on the Navibar. The second section of the document demonstrates control over the pictures using a separate set of buttons.

When the `navibar` key is not present, then control over the graphics and captions passes to a predefined set of buttons. These buttons are

```
\togglePrev[⟨eforms_opts⟩]{⟨baseName⟩}{⟨width⟩}{⟨height⟩}{⟨nPics⟩}
\toggleNext[⟨eforms_opts⟩]{⟨baseName⟩}{⟨width⟩}{⟨height⟩}{⟨nPics⟩}
\toggleClear[⟨eforms_opts⟩]{⟨baseName⟩}{⟨width⟩}{⟨height⟩}
\togglePlay[⟨eforms_opts⟩]{⟨baseName⟩}{⟨width⟩}{⟨height⟩}{⟨nPics⟩}{⟨play_time⟩}
\toggleStop[⟨eforms_opts⟩]{⟨baseName⟩}{⟨width⟩}{⟨height⟩}
\toggleSpeedControl[⟨eforms_opts⟩]{⟨baseName⟩}{⟨width⟩}{⟨height⟩}{⟨nPics⟩}
\toggleReset[⟨eforms_opts⟩]{⟨baseName⟩}{⟨width⟩}{⟨height⟩}
```

Command Location: Used within the `slide` environment.

Parameter Description: Each of these have a number of parameters.

1. `eforms_opts`: This optional argument can be used to change the appearance of the buttons. See the `eforms` Package Documentation for details.
2. `baseName`: Common name of all toggle fields to be in the same group.
3. `width`: The width of the button, or in the case of `\toggleSpeedControl`, which is a combo box (dropdown menu), the width of the combo box. In the case of a button, if this argument is left empty, the width will be calculated automatically based on the text that is to appear on the button.
4. `height`: The height of the button or combo box.
5. `nPics`: The number of pictures/captions in this toggle field.
6. `play_time`: The time in seconds to show each picture, used if combo box is not present.

ex9 Example 60: See `apb_ex9.tex` for an example of the use of these buttons.

Finally, we mention `\disableToggleControls` for disabling the button controls just discussed, when the document is in *full screen mode*. When a slide show is active, this prevents someone from clicking on the buttons for the graphics/captions show.

```
\disableToggleControls
```

Command Location: Use this command in the preamble.

14.3. The `\slideshow` Option

When the document is compiled with the `slideshow` option, the ‘FS’ and ‘SlideShow’ buttons (see Table 2, page 43) appear on the first page. Clicking on the ‘SlideShow’ button puts the document into full screen mode,¹¹ and starts a code that programmatically presses the ‘Next’ button. (Actually, it executes the same JavaScript function that is executed when the ‘Next’ button is physically pressed.)

```
\SlideshowTiming{⟨time_length⟩}
```

Command Location: Place this command in the preamble.

¹¹The document must be viewed using Adobe Reader or **Acrobat**, outside of a Web browser.

Parameter Description: The programmatic execution of the ‘Next’ button occurs at the end of each time interval of length determined by the argument of `\SlideshowTiming`. The argument `time_length` is measured in seconds, with a default of `\SlideshowTiming{3}`. Thus, unless the value is changed, a new “talking point” will appear every 3 seconds.

ex10 **Example 61:** The tutorial file `apb_ex10.tex` demonstrates the basic functionality of the `slideshow` option.

- **A Graphics/Captions Slide Show Compatibility**

In the paragraphs on ‘[Creating a Graphics/Captions Slide Show](#)’ on page 73, we introduced the Graphics/Captions slide show feature of APB. This slide show is compatible with a *document slide show*, that is, a document compiled with the `slideshow` option.

ex11 **Example 62:** The tutorial file `apb_ex11.tex` demonstrates how to set up a document slide show that only has graphics/captions slide shows and no “talking points”.

- **A Graphics/Captions Slide Show with Talking Points**

Yes, “talking points” can easily be mixed in with a graphics/captions slide show.

ex12 **Example 63:** The tutorial file `apb_ex12.tex` demonstrates how to set up a document slide show that has both talking points and graphics/captions shows.


14.4. The APB Help Feature


The APB provides two methods of creating help screens or frames, one using layers, the other using the tool tip feature of **Acrobat** Forms. The command names for these two methods are `\texHelp` and `\pdfHelp`, respectively.

```
\texHelp{\layer_name}{\text}
```

Command Description: This command takes three parameters:

1. `layer_name`: The name of a layer created by `\xBld` containing the help text.
2. `text`: The text that the help is attached to.


 This is a very powerful method, the content of the help screen (or frame) can be any typeset material, including mathematics, graphics, and even animations.

 The file `apb_help.tex` is a tutorial on the use of this feature. The file `apb_help1.tex` gives a more complex example using `PSTricks` and `\fBld`.

```
\pdfHelp{\name}{\help_tip}{\text}
```

Command Description: This command takes three parameters:

1. `name`: The field name of the underlying **Acrobat** button used to create to the tool tip.
2. `help_tip`: The actual text that will be the tool tip.
3. `text`: The text that the help is attached to.

 The file `apb_help.tex` is a tutorial on the use of this feature.

15. Printing the Presentation

The APB can be used to create a beautiful (or a plain) document for an invited talk, for the classroom, or for distribution over the Internet. An APB document is primarily digital—replete with “talking points” and animations—, meant to be viewed in the Adobe Reader 7.0 or later, and not meant to be printed. Yet, there is the occasion in which it is desired to have a printed version of the presentation. In this section, we discuss methods of creating a print version of an APB document.

15.1. Printable Copy of the Presentation

The web package has two print options, `forpaper` and `forcolorpaper`. To create a printable document from an APB document, re-compile using one of these two options. But wait, it's not as simple as that! Early in the development of the APB, it was decided that a printable copy of an APB document is for printing! Therefore, when the document is compiled with one of the two paper options the following changes are made to the document:

1. Layers are not created, “talking points” appear as text, for example.
2. The Navibar is removed (it's not needed for a print version of the document, just flip through the paper pages in the usual way).
3. If a panel option is used, the navigation panel and the outline of the talk (generated by the `paneloutline` option of APB) is also removed. When the outline panel does not appear the value of `\altniddlepanel` is used in its place.
4. Inclusion of the document JavaScript is turned off. (No layers, no Navibar, no need for the JS of APB.)
5. The `slide` environment no longer starts a new page, hence, text is allowed to flow across pages, but does insert the text ‘Slide n ▼’ at the beginning of each slide (except for the first one), where ‘n’ is the slide number. The content of the text is contained in the text macro `\apb@newslide@text`, defined as follows:

```
\newcommand{\apb@newslide@text}{%
  \makebox[0pt][r]{Slide~\theocslide\ $\blacktriangledown$}}
```

The `\apb@newslide@text` can be redefined (using `\renewcommand`) to suite your needs.

The `slide` environment maintains a counter `ocslide` that counts the number of each slide.

6. The `\textwidth` is preserved, so the document (should) still have the same line breaks, while `\textheight` is set back to its default value according to the paper size.

ex15 **Example 64:** `apb_ex15.tex` is a tutorial on how to create a document for both presentation and for print.¹²

The following are some thoughts on designing a document for print.

1. Use the `forscreen` and `forpaper` environments, Section 5.8, to enclosed commands (in the preamble, between slides and within slides) and content (within a slide) that are appropriate to the screen presentation and appropriate for the paper document.

Some commands, such as `\importSounds`, are clearly directed at a screen presentation and should be enclosed in the `forscreen` environment in the preamble.

2. Remove the background graphics for the paper version. The `\ClearTemplatesOnly` command is a convenience command defined in APB for that purpose, it removes the graphics but preserves the background colors,

```
\ClearTemplatesOnly \ClearAllTemplates
```

The `\ClearAllTemplates` clears the same templates as `\ClearTemplatesOnly`, but also clears the background colors.

Thus, in the preamble of the document, we could have,

¹²These considerations are not necessary if the presentation is not to appear in paper form.

```
\begin{forpaper}  
\ClearTemplatesOnly  
\end{forpaper}
```

which would clear away the background graphics from the screen, the panel and the full width screen.

3. The theme backgrounds (accessed through the `theme` option) are all enclosed in a `forscreen` environment. These are designed for the `designv` design option of `Web`, and would not look good for any other paper sizes. The backgrounds obtained from the `apbBg` option are not enclosed in the `forscreen` environment and would be available in the `forcolorpaper` option. These can be removed using the `\ClearTemplatesOnly` or the `\ClearAllTemplates` commands.
4. Additions templates or watermarks created using `\AddToTemplate` and `\AddToPanelTemplate` should be enclosed in a `forscreen` environment unless you intend to use them for paper or have no intention of creating a paper version of the document.
5. Any transition effects or open or close page actions should be enclosed in the `forscreen` environment (unless, there is no intent to create a paper version, of course).
6. The `annot`, `authorannot` and `publ` cannot environments, discussed in the next section should not be inside a `forscreen` or a `forpaper` environment. They take care of themselves.

15.2. Thumbnails of the Presentation

Another way of representing an APB document on paper is through a thumbnail document, each thumbnail is a snapshot of the presentation pages. This document can be printed and distributed to the target audience.

There are two thumbnail functions available through APB and **Acrobat Pro 7.0** or later. The first is the simplest, the thumbnails are created by printing “two-up” through **Acrobat**; the second method is more interesting, here you can create your thumbnails (using an entirely different method) and associate annotations, comments or notes. These two methods are discussed in the next two sections.

- **Create Thumbnails (no annots)**

Develop the APB presentation document, and when satisfied, create a simple thumbnail document as follows:

1. Bring the document into **Acrobat Pro 7.0** or later.
2. Once your file is open in **Acrobat**, go to the menu

`Tools > AcroTeX Presentation Bundle > Show All Layers.`

for **Acrobat Pro 9** or prior, and to

`View > AcroTeX Presentation Bundle > Show All Layers.`

for **Acrobat Pro X** or later.

This programmatically shows all the layers. The console window should pop up with additional instructions.

(APB 2.0) The use of the `Show All Layers` is really not needed because the common build commands are printable, even if they are hidden. You can design your anime so the last one in the sequence is printable, while the others are not (by default).

3. Before you make a thumbnail image of each page, page through your document and make any final adjustment to the pages (what you see is what you get); and run your animations, if any, to display the final image.

4. While your PDF document is still in the viewer—and it should be the *only file open in the viewer*—, go to the menu

Tools > AcroTeX Presentation Bundle > Create Thumbnails (no annots).

for **Acrobat Pro 9** or prior, and to

View > AcroTeX Presentation Bundle > Create Thumbnails (no annots).

for **Acrobat Pro X** or later.

This menu item executes some JavaScript that prints the document in a “two-up” format. You will be prompted for a file path to save the newly created document.

- **Create Thumbnails (annots)**

This feature of APB may be somewhat unique among L^AT_EX-based presentation packages. As in the previous section, we create thumbnails of the presentation, but unlike the previous technique, we can build a document that displays these thumbnails along with associated comments of the author.

ex16 **Example 65:** The main working example and tutorial on this topic is `apb_ex16`. See this file for the details presented here, and maybe a little more.

The feature of writing an annot file is activated when the `annotslides` option is taken. This option takes one of four values:

1. `nonotes`: The file `\jobname_thmbnonotes.ltx` is programmatically generated. This file produces two-up thumbnails of the APB document, with optional captions under each thumbnail. The captions are extracted from the content of the `annot` environment placed throughout the source file of the presentation. See the left figure in [Figure 8](#).
2. `notes`: The file `\jobname_thmbnotes.ltx` is created in this case. The file produces the thumbnails of the presentation in the left column, with the captions optionally placed beneath them. In the right column is a rectangular region for the audience members to take notes on the presentation, across from the thumbnails. The caption is extracted from the content of the `annot` environment placed throughout the source file of the presentation. See the left figure in [Figure 8](#).
3. `forauthor`: The file `\jobname_thmbauthor.ltx` is generated. This file produces the thumbnails on the left, and author’s comments on the right. The comments of the author are based on the contents of the `authorannot` environment placed throughout the source file of the presentation. Again, the caption can be optionally included. See the right figure in [Figure 9](#).
4. `forpublic`: The file `\jobname_thmbpublic.ltx` is programmatically generated. This file produces the thumbnails on the left, and author’s comments for the public on the right. The comments meant for the public are based on the contents of the `publicannot` environment placed throughout the source file of the presentation. Again, the caption can be optionally included. See the right figure in [Figure 9](#).

- As suggested in the list above, the captions can be optionally included in each of the four generated files. The captions appear by default, but should you want to exclude the captions, use the APB option `excludeannots`.

As you create your beautiful presentation full of style and content, think about adding a few notes for yourself and for the audience. APB defines three environments to be used between slides:

Section 15: Printing the Presentation

```
\begin{annot}
...
\end{annot}

\begin{authorannot}
...
\end{authorannot}

\begin{publicannot}
...
\end{publicannot}
```

- The `annot` environment is used to write short descriptions (captions, if you will) for each slide. The `annot` environment should follow the slide that it describes.
- The `authorannot` environment is used to by the document to write notes to himself for reference during the presentation. (Presentation notes for the author.) This environment should follow the `annot` environment, if present.
- The `publicannot` environment is used to write comments the document author wants the public (the audience) to know about each slide, beyond what is on the slide. This is a good place to put in email addresses and URLs. This environment should follow the `annot` environment, if present.

• The `annot` environment *must be placed before* the `authorannot` and `publicannot` environment, the latter two can appear in any order. This restriction is necessary to get the contents of the environments written to the generated file in the correct order.

```
\begin{slide}
Some real neat stuff for everyone to know...
...
\end{slide}

\begin{annot}
This slide shows some real neat stuff that
everyone should know.
\end{annot}
```

The contents of the `annot` environment appear under the thumbnail of the associated slide, when the value of the APB option `annotslides` is `nonotes` or `notes`. [Figure 8](#) shows the position of the content in the thumbnails document.

Setting `annotslides=author` in the option list of APB enables you to create a thumbnail document containing comments the author might want to make during the course of the presentation. [Figure 9](#) shows the thumbnail document created with this option choice. The author makes comments in the `authorannot` environment, like so...

Section 15: Printing the Presentation

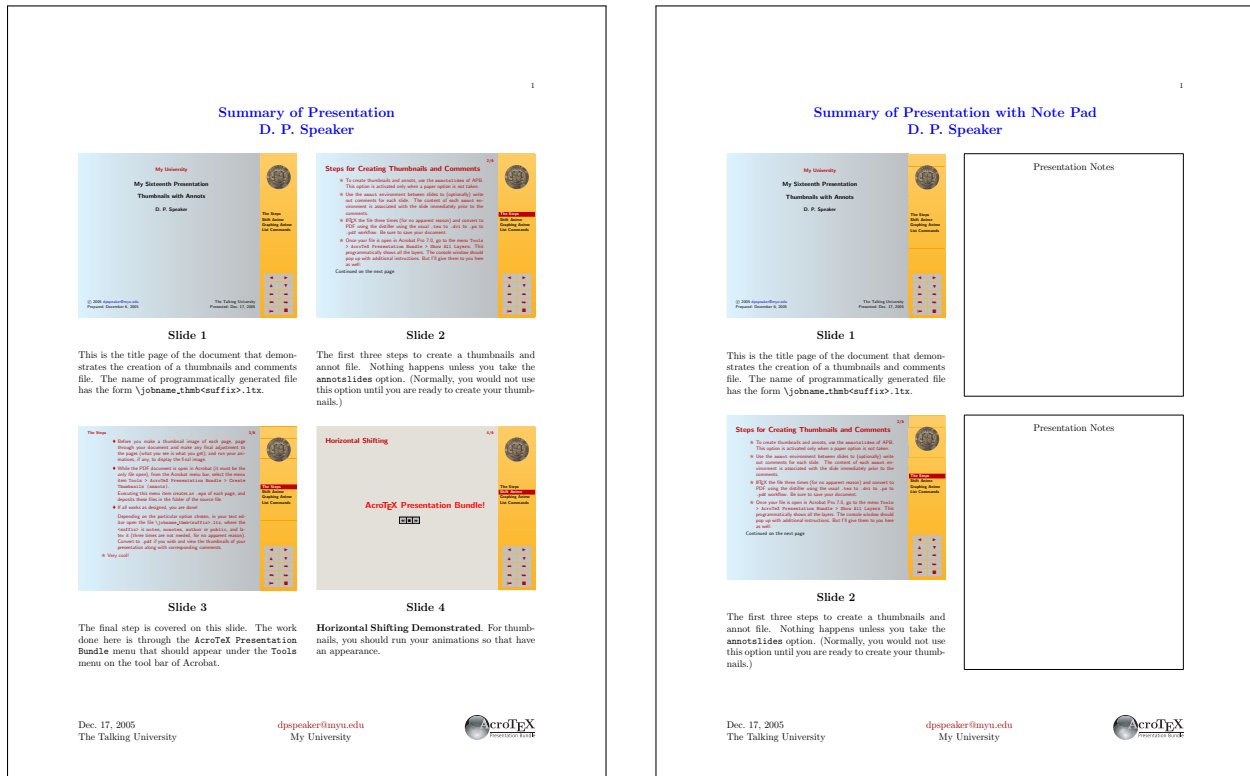


Figure 8: Thumbnail Documents: `annotslides=nonotes` and `annotslides=notes`

```

\begin{slide}
Some real neat stuff for everyone to know...
...
\end{slide}

\begin{annot}
This slide shows some real neat stuff that
everyone should know.
\end{annot}

\begin{authorannot}
Major points to be made on this page:
\begin{enumerate}
\item Discuss each item point by point,
\item Don't skip over any points.
\item Be sure to turn on your computer, very embarrassing otherwise.
\end{enumerate}
\end{authorannot}

```

When `annotslides=forpublic`, the contents of the `publi` cannot environments are used; the contents of the environments contain comments that will appear to the right of the thumbnail of the slide. Include the major points of each slide, and anything not explicitly appearing in a slide, such as a mailing address, email address, URLs, and other technical comments. Figure 9 shows the thumbnail document created with this option choice.

How do you create these amazing thumbnail and annot files, you say? Well, I'll tell you now.

Section 15: Printing the Presentation

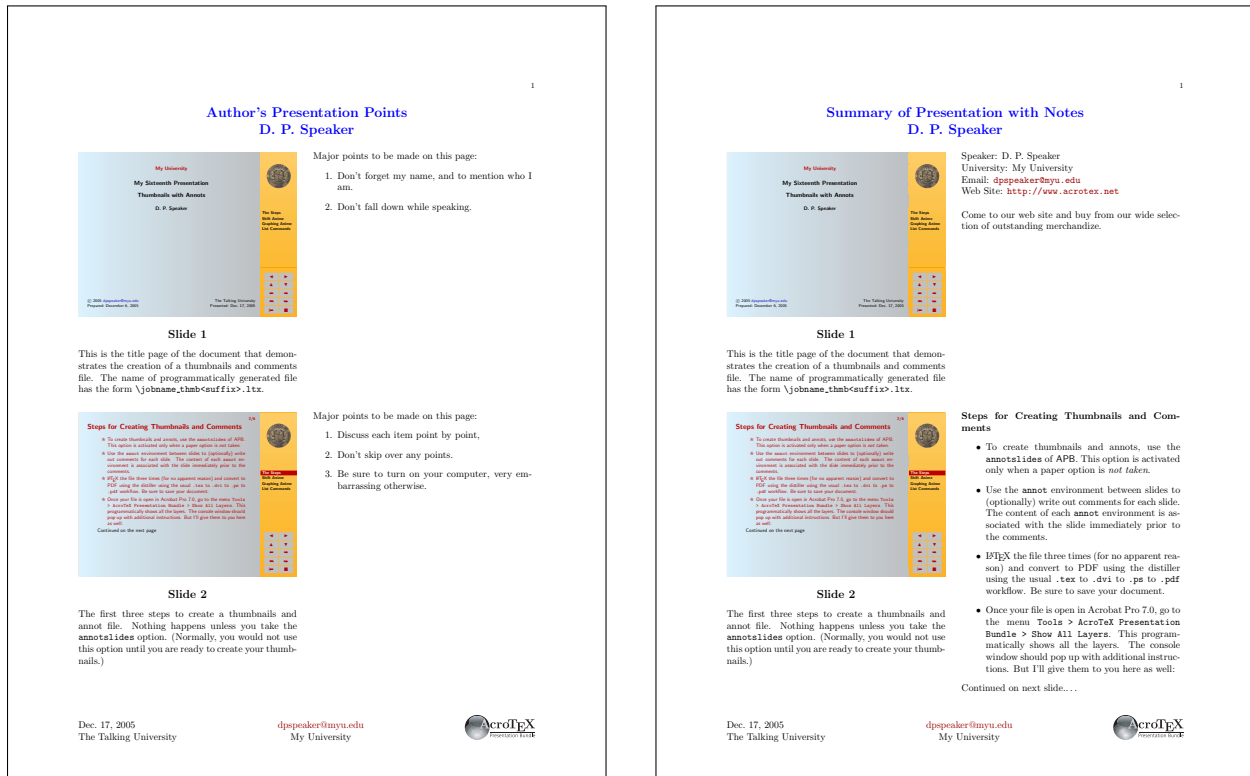


Figure 9: Thumbnail Documents: `annotslides=forauthor` and `annotslides=forpublic`

1. To create thumbnails and annots, use the `annotslides` of APB. This option is activated only when a paper option is *not taken*.
2. Use the `annot` and `authorannot` environments between slides to (optionally) write out comments for each slide. The content of each `annot` environment is associated with the slide immediately prior to the comments.
3. \LaTeX the file three times (for no apparent reason) and convert to PDF via the distiller using the usual `.tex` to `.dvi` to `.ps` to `.pdf` workflow. Be sure to save your document.
4. Once your file is open in **Acrobat**, go to the menu

Tools > AcroTeX Presentation Bundle > Show All Layers

for Version 9 or prior, and to

View > AcroTeX Presentation Bundle > Show All Layers

for Version 10 or later.

This programmatically shows all the layers. The console window should pop up with additional instructions. But I'll give them to you here as well:

- (a) Before you make a thumbnail image of each page, page through your document and make any final adjustment to the pages (what you see is what you get); and run your animations, if any, to display the final image.
- (b) While your PDF document is still in the viewer—and it should be the *only file open in the viewer*—, go to

Tools > AcroTeX Presentation Bundle > Create Thumbnails (annots)

Section 15: Printing the Presentation

for Version 9 or prior, and to

View > AcroTeX Presentation Bundle > Create Thumbnails (annots)

for Version 10 or later.

Executing this menu item creates an .eps of each page, and deposits these files in the folder of the source file.

(c) If all works as designed, you are done!

Depending on the option chosen (nonotes, notes, forauthor, forpublic), open the file `\jobname_thmb<suffix>.ltx` (where the `<suffix>` is nonotes, notes, author or public, respectively) in your editor, and latex it (three times are not needed, for no apparent reason). Convert to PDF if you wish and view the thumbnails of your presentation along with corresponding comments, or print for distribution to the audience.

(APB 2.0) The use of the Show All Layers is really not needed because the common build commands are printable, even if they are hidden. You can design your anime so the last one in the sequence is printable, while the others are not (by default).

• Customizing the Thumbnail/Annot Generated File

When `annotslides` option is used, APB writes the file `\jobname_thmb<suffix>.ltx` (where `<suffix>` = nonotes, notes, forauthor, forpublic). Some control over the creation of this file is available in the preamble of the source file `\jobname.tex`. Use the `annoheader` and the `annotpreamble` environments to write the document header and the document preamble of the generated thumbnail file.

```
\begin{annoheader}
...
\end{annoheader}
```

Use this environment in the preamble of the source file to write a header for the thumbnail file (here, the header is a listing of the class and packages for the file). If the `annoheader` environment is not present in the source file, APB uses the following header:

```
\begin{annoheader}
\documentclass{article}      % Add in options, or possibly another class
\usepackage{color}          % required can include options
\usepackage{graphicx,calc}   % required
\usepackage{web}            % Optional, but highly recommended
\margins{1in}{1in}{1in}{1in} % required if web is used, but can
\end{annoheader}            % change dimensions
```

Usually, the `annoheader` environment need not be included in the source file.

The `annoheader` environment also secretly includes the `apbthumb` package, part of the distribution of APB. The `apbthumb` package is used to provide some basic formatting for the thumbnail file.

```
\begin{annotpreamble}
...
\end{annotpreamble}
```

The `annotpreamble` environment is used to enter any additional packages or command definitions beyond what is introduced through the (default) `annoheader` environment.

The `annotpreamble` environment *must follow* the `annoheader` environment, if present.

Below is an example of an `annotpreamble`. When the `web` package is used (the default, highly recommended), the `apbthumb` package recognizes several commands for entering document info. In addition to all document info commands recognized by `web` (these are `\title`, `\author`, `\subject`, `\email`, `\keywords`, `\university`, `\version`, `\copyrightyear`).¹³ In addition to these, the command `\talksite` is also recognized (this has the same meaning as in the APB).

There are four new title commands, one each for the four versions of the thumbnail file; These are `\nonotestitle`, `\notestitle`, `\authortitle` and `\publictitle`. These are the titles used for each of the four thumbnail files, corresponding to the `annotslides` values of `nonotes`, `notes`, `forauthor`, `forpublic`, respectively. Each of these specialized titles have an optional parameter entering the short version of the title. The short version is used as the running header of the thumbnails file. (Naturally, if there is no short title, the title is used.

The `apbthumb` package checks whether the appropriate special title is present, if it is, that title is used; if not, it checks for the presence of the `\title` command, if it is, that title is used; finally, if `\title` is not present, a default title (and short title) is used.

ex16 Example 66: In this example, we include titles for each of the four options for `annotslides`, as well as other document info commands. There is also a command definition needed in the file.

```
\begin{annotpreamble}
\nonotestitle{Summary of Presentation}
\notestitle{Summary of Presentation with Note Pad}
\authortitle[Author's Talking Points]{Author's Presentation Points}
\publictitle{Summary of Presentation with Notes}
\date{Dec.\ 17, 2005}
\talksite{The Talking University}
\author{D. P. Speaker}
\email{dpspeaker@myu.edu}
\university{My University}

% A macro definition for \cs used in the comments.
\newcommand{\cs}[1]{\texttt{\char'\#1}}
\end{annotpreamble}
```

- The `apbthumb` package has additional features that can be used to customize the thumbnail files. See the `apbthumb` package [Documentation](#) for additional details and features not mentioned here.

16. APB and Exerquiz

The `APB` works well with the [AcroT_EX eDucation Bundle \(AeB\)](#), in particular, `Exerquiz` works as advertised from within `APB`.

- If you want to use `Exerquiz`, load this package before loading `APB`. The recommended sequence for loading is `Web`, `Exerquiz`, and `APB`, in that order.

The `APB` makes some definitions so that the elements of exercises and quizzes can be stepped through as “talking points”; other definitions were needed to enable the solutions to be displayed properly (remember, all slides must be enclosed in a `slide` environment). This section briefly discusses those definitions, gives some simple examples, and illustrates tricks and techniques developed.

- The folder `examples/apb_aeb` contains several files that illustrate some of the techniques discussed in this section.

16.1. Exerquiz Solutions

In this section we discuss special commands and environments for handling the solutions to exercises and quizzes created by `AeB`. The solution pages are a bit of a problem because they cannot be directly accessed by the document author, they are generated by the `LATEX` code in the `Exerquiz` package.

¹³Some of these are not used, but recognized. Others are placed in the Document Info fields of the PDF.

- **The Title Pages for Solutions**

There are two sets of solutions, one for exercises and one for quizzes. The first solution normally lies on the title page, the page with the section title; however, should you want to have some text or graphics immediately following the section title, enter that content into the `extitlepage` or `quiztitlepage` environment, as appropriate.

```
\begin{extitlepage}
...
<Text for the title page of solutions to exercises>
...
\end{extitlepage}

\begin{quiztitlepage}
...
<Text for the title page of solutions to quizzes>
...
\end{quiztitlepage}
```

Environment Location: These environments can go anywhere outside a slide environment; however, the recommended location is in the *preamble of the document*.

- Should one of these environments appear (in the preamble), the content of that environment appears on the title page of the corresponding solution set; in this case, the first solution begins on the next page. If one of these environments does not appear (in the preamble), the first solution for that solution set begins on the title page, which is the default for AeB.

- **Multi-page Solutions**

Recall that in APB there is no text flow from one page to the next. This is a problem if the solution to a quiz or exercise has more content that can be held on a single page; there is a simple fix for this, however. APB makes definitions what wrap each solution in a `slide` environment. These definitions begin a slide at the beginning of the solution and end a slide at the end of the solution. This makes it easy to break up a solution, just insert a `\end{slide}` and a `\begin{slide}` were appropriate, as the following code snippet illustrates:

Example 67: Introduce an `\end{slide}` and a `\begin{slide}` from within a solution.

```
\begin{solution}
This is a long solution to a quiz or exercise
...
...
\end{slide}
\begin{slide}
The solution continues on this next page...
...
\end{solution}
```

Though the environment delimiters appear to be unbalanced, they are in fact not; `\begin{solution}` generates `\begin{slide}` and `\end{solution}` generates an `\end{slide}`.

- **The `apbwriteto` Environment**

As explained in earlier, between slides you can insert non-typeset material such as commands to remove or change backgrounds, or special effects such as a page open or page close action. To do this for the solutions, you have to write to the appropriate solutions file, use the `apbwriteto` environment for this purpose.

```
\begin{apbwriteto}{\soln_file}
...
⟨commands and/or environments⟩
...
\end{apbwriteto}
```

Command Location: Meant to be used just above the `\begin{solution}` of the Exerquiz package if the commands effect the next solution, or just after `\end{solution}` for annotating that solution slide using the `annot`, `authorannot`, or `publicannot` environment. See ‘[Create Thumbnails \(annots\)](#)’ on page 79 for the descriptions of these environments and their use.

Parameter Description: The `\soln_file` parameter must be either `ex` to write to the file containing the solutions to exercises, or `quiz` to write to the solutions to the quizzes file. The content parameter contains the commands and environments to be written to the solutions file.

Example 68: Example of usage of the `apbwriteto` environment and assumes knowledge of Exerquiz quizzes. Here, we clear all underlying graphic templates.

```
\item $\cos(\pi) = \RespBox{-1}*{1}{.0001}[[2,4]]\kern1bp\CorrAnsButton{-1}$

% Clear all templates before this solution
\begin{apbwriteto}{quiz}
\ClearTemplatesOnly
\end{apbwriteto}
\begin{solution}
Everyone knows, and you should too, that  $\boxed{\cos(\pi) = -1}$ .
\end{solution}
```

- **Setting Slide Options**

The final topic in this section concerns introducing the slide options into the slide environment surrounding the solutions to exercises and quizzes. Changes in the slide options can be done globally or locally.

For a global change, use `\setDefaultSlides`, as described in [Section 9.5](#), titled “Set Default Slide Options”. This command can be placed just above `\end{document}` and would effect all the solution slides which are generated at the end of the document.

For a local change, use the `\setLocalSlideOpts` command within the `apbwriteto` environment just above the target slide.

```
\begin{apbwriteto}{\soln_file}
...
\setLocalSlideOpts{\slide_key-value_pairs}
...
\end{apbwriteto}
\begin{solution}
...
\end{solution}
```

Command Location: As stated above, use `\setLocalSlideOpts` within the `apbwriteto` environment just above the target slide. The `\setLocalSlideOpts` command is designed for use only for solutions slides, and should not be used otherwise.

Parameter Description: The `slide_key-value_pairs` parameter is a comma-delimited list of key-value pairs, as documented in [Section 7.1](#), “The slide Environment”, on page 26.

Example 69: In the example below, we save the graphical elements currently in effect, clear all templates and feed the slide environment for the next solution some key-value pairs. After the end of the solution, we restore the graphical layout to the state it was in before the solution.

```
\begin{apbwriteto}{quiz}
\saveElements{myStuff}
\ClearTemplatesOnly
\setLocalSlideOpts{fontseries=bfseries,color=red,fontsize=Large}
\end{apbwriteto}
\begin{solution}
Is this bold, red and Large, or what?
\end{solution}
% Now, let's restore the elements for his next slide.
\begin{apbwriteto}{quiz}
\restoreElements{myStuff}
\end{apbwriteto}
```

- There is one exception to the use of `\setLocalSlideOpts` within the `apbwriteto` environment. For technical reasons, inserting slide options for one of the title pages for the solution sets is a special case. For this purpose use the following commands.

```
\setSlideOptsEx{<slide_key-value_pairs>}
\setSlideOptsQuiz{<slide_key-value_pairs>}
```


Command Location: Place these commands in the preamble.

16.2. Stepping Through an Exercise or Quiz

The APB and AeB have great potential as teaching tools. Teachers can write short teaching materials using APB and include little exercises and/or quizzes to test the students’ understanding. In this section we discuss techniques for using AeB with APB. Note, no special techniques are needed if you do not want to create “talking points” build around the various components of the exercises and quizzes of AeB.

- **Exercises**

The exercise environment of the [AcroT_EX eDucation Bundle \(AeB\)](#) is the easiest case. The normal use of any of the “talking point” commands (`\B1d`, `\fB1d` and `\bB1d` to name a few of these) will work.

-  **Example 70:** The example file `apb_aeb.tex` illustrates how to create talking points with the exercise environment. See also the folder `examples/apb_aeb` for many additional examples.

- **Forms in General**

Forms are not considered content in the PDF specification. Form fields are laid on top of the content; therefore, they are not part of any optional content (OCG or layer).

The scheme devised to give the appearance that the fields are part of the layers is to associate a JavaScript action to the layer that hides or displays the field or fields as the layer is hidden or displayed. The two commands that have this feature are `\fB1d` (a SimpleOC) and `\bB1d` (a Nested OC).

The SimpleOC might be easier to use in most situations. Just before your content and field(s), place an `\fB1d` command with an optional key-value pair of the `fields` key and a list of your fields. For example.

```

\FBld[fields="myField1"]{myField1}
!ameta( content (and field) )

\FBld[fields={"myField2","myField3","myField4"}]{myFields}
!ameta( content (and field) )

\FBld[fields="myField1",sound=trek.wav]{myFieldandSound}
!ameta( content (and field) )

```

The first example is for a single field. In the second example, several fields are included so they must be enclosed in matching braces and separated by commas. In the third, we have a single field and a sound associated with this layer. All form fields must be enclosed in matching double quotes. The fields do not have to be within the scope of the content for `\FBld`. See `\FBld` on page 59 for more details of the `\FBld` command.

Similarly for `\bBld`, we reproduce only the second example above.

```

\bBld[fields={"myField2","myField3","myField4"}]{myFields}
!ameta( content (and field) )\eBld

```

The rules are the same as for `\FBld`. Note that for `\bBld` the content is delimited by `\eBld`. Again, the forms references do need not be in the designated content. See `\bBld` on page 61 for more details of the `\bBld` command.

- The field names and the layer names are written to the `\jobname.aux` file and read back in at the beginning of the file. This information is then written to the hard drive again as part of some JavaScript code. After the file is built into a PDF, each time the file is opened some startup JavaScript is run to assign the actions described above.

In the next two sections the `shortquiz` and `quiz` environments are discussed. In these environments, the names of the form fields are assigned automatically by \TeX at compile time. Knowledge of these field names is essential to correctly assigning the values of the `fields` key for `\FBld` and `\bBld`.

• Short Quizzes

A `shortquiz` environment creates a quiz with instant feedback. When the user enters a response to a question, the user is informed immediately on the correctness of the response. Questions can be multiple choice or fill-in the blank. In addition there are a number of supporting form fields, a button to get the correct answer or to jump to the solution; tally fields to keep a running total of missed questions; a clear all fields button. To control the form elements of the short quiz, a knowledge of the field names is necessary.

When a `shortquiz` environment is begun, there is an optional argument for entering the base name of the quiz. This base name is used to build the field names of all form elements used in the quiz. The base name is saved in the macro `\currQuiz`. Short quiz uses the counter `questionno` to index the questions as well.

The field names are of two types: Names that refer to a form element of a question and names that refer to a form element of the quiz. The syntax of these two types are

```

<suffix>.\currQuiz.\thequestionno
<suffix>.\currQuiz

```

respectively.

Form Elements for the Quiz Questions		
Description	Command(s)	Suffix
Multiple Choice Question	<code>\Ans</code>	mc
Fill-in Question	<code>\RespBoxMath</code> , <code>\RespBoxTxt</code>	obj
Grouped Responses	<code>\begin{mathGrp}... \end{mathGrp}</code>	grpobj

Description	Form Elements for the Quiz Command(s)	Suffix
Tally Box	\sqTallyBox	tally
Correction Button	\CorrAnsButton,\CorrAnsButtonGrp	corr
Form Elements for the Quiz		
Clear Button	\sqClearButton	clear
Total Tally	\sqTallyTotal	tallytotal

Helper commands have been defined to make it easier to input the value of the name of the field correctly.

```
\qNameN{<suffix>}
\qNameRN{<suffix>}{<number>}
\qNameAN{<suffix>}{<number>}
\qName{<suffix>}
```

Command Location: These commands are used as the value(s) of the `fields` key of `\fBld` or `\bBld`.

Command Description: In many cases the `\fBld` or `\bBld` command appears just before the question and the field that is the value of the `fields` key. At this point, the counter `questionno` has not been incremented yet.

- `\qNameN`: Name for the form element of the next question. The required parameter for this command is the appropriate suffix for the form element. The definition for this command is

```
\newcommand\qNameN[1]{"#1.\currQuiz."+(\thequestionno+1)}
```

- `\qNameRN`: Name for the form element relative to the current one. The command with parameters `\qNameRN{<suffix>}{1}` is the same as `\qNameN{<suffix>}`, while `\qNameRN{<suffix>}{2}` is the form field following the next one. The definition for this command is


```
\newcommand\qNameRN[2]{"#1.\currQuiz."+(\thequestionno+#2)}
```

- `\qNameAN`: Name for the form element of a question. The second parameter is a number. Used as an absolute reference to the field, as opposed to a relative reference. The definition for this command is

```
\newcommand\qNameAN[2]{"#1.\currQuiz.#2"}
```

- `\qName`: Name for quiz support field elements. The parameter is the suffix for the support element (`clear` and `tallytotal`). The definition for this command is

```
\newcommand\qName[1]{"#1.\currQuiz"}
```

 **Example 71:** The example file `apb_aeb.tex` contains extensive examples of the usage of these commands. See also the folder `examples/apb_aeb` for many additional examples.

• Quizzes

A quiz environment creates a quiz with delayed feedback. When the user enters a response to a question, the response is parsed for syntax errors. Questions can be multiple choice or fill-in the blank. In addition there are a number of supporting form fields, a button to get the correct answer or to jump to the solution; tally fields to keep a running total of missed questions; a clear all fields button. To control the form elements of the quiz, a knowledge of the field names is necessary.

When a quiz environment is begun, there is a required argument for entering the base name of the quiz. This base name is used to build the field names of all form elements used in the quiz. The base name is saved in the macro `\currQuiz`. The quiz environment uses the counter `questionno` to index the questions as well.


As with the short quiz, the field names are of two types: Names that refer to a form element of a question and names that refer to a form element of the quiz. The syntax of these two types are

```
(suffix).\currQuiz.\thequestionno
(suffix).\currQuiz
```

respectively.

Form Elements for the Quiz Questions		
Description	Command(s)	Suffix
Multiple Choice Question	\Ans	mc
Fill-in Question	\RespBoxMath,\RespBoxTxt	obj
Grouped Responses	\begin{mathGrp}...\end{mathGrp}	grpobj
Correction Button	\CorrAnsButton,\CorrAnsButtonGrp	corr
Prompt Button	\PromptButton	promptButton
Form Elements for the Quiz		
Begin Quiz	\begin{quiz}	beginQuiz
End Quiz	\end{quiz}	endQuiz
Answer Field	\AnswerField	Ans
Score Field	\ScoreField	ScoreField
Grade Field	\GradeField	GradeField
Points Field	\PointsField	PointsField
Percent Field	\PercentField	PercentField
Corrections Button	\eqButton	correct


The helper commands `\qNameN`, `\qNameRN`, `\qNameAN` and `\qName`—described in the section ‘[Short Quizzes](#)’ on page 88—are used to reference these field names in the `\fBld` and `\bBld` commands.

 **Example 72:** The example file `apb_aeb.tex` contains extensive examples of the usage of these commands. See also the folder `examples/apb_aeb` for many additional examples.

16.3. Crossing Slide Boundaries

All the benefits of APB come with another set of problems; crossing slide boundaries is one of them. Because a `slide` environment begins a group, any other group that begins in one slide cannot continue to the next slide, without closing its group. The `exercise`, `shortquiz` and `quiz` environments all begin groups. Here lies our current problem.

Commands have been developed to enable you to cross slide boundaries, they seem to work properly, bar any unforeseen circumstances. Let us take each of the three environments in turn.

 **Example 73:** The example file `apb_aeb_xslide.tex` contains examples to illustrate the techniques and commands covered in the next paragraphs.

- **The exercise Environment**

No special treatment is needed for exercises *without parts* (with or without solutions after). A slide should be well designed, a single exercise of this type should be fit on one slide. For an exercise with parts, there could be many parts, so many as to not fit conveniently on one slide.

```
...(exercise_environment)
\pushExercise{(exercise_env_name)}
...
\end{slide}
\begin{slide}
...
\popExercise{(exercise_env_name)}{(env_args)}
...(exercise_continues)
```

Command Location: Place the `\pushExercise` command between items,¹⁴ within the parts environment, of a multi-part exercise.

¹⁴Between parts started by the `\item` command, and following the `\end{solution}` of the previous item, if there is a previous item, if there is a solution.

Command Description: An exercise environment may be used to define a problem or an example environment.¹⁵ The parameter $\langle exercise_env_name \rangle$ is the name of the exercise environment, usually, this will be `exercise`. For the `\popExercise`, the first argument is the same $\langle exercise_env_name \rangle$ that was pushed earlier, the second argument $\langle env_args \rangle$ is a copy of the parameters used when the original environment was initiated.

- **The shortquiz Environment**

The `shortquiz` environment can be handled easier than exercises. There are two commands only, with no parameters, they are `\pushShortQuiz` and `\popShortQuiz`.

```
...
\shortquiz
\pushShortQuiz
...
\end{slide}
\begin{slide}
....
\popShortQuiz
\shortquiz-continues
```

Command Location: Place `\pushShortQuiz` between items,¹⁶ within a questions environment, of a multi-question short quiz.

- **The quiz Environment**

The `quiz` environment can be broken between two slides in the same way as the `shortquiz` environment.

```
...
\quiz
\pushQuiz
...
\end{slide}
\begin{slide}
...
\popQuiz
...
\quiz-continues
```

Command Location: Place `\pushShortQuiz` between items,¹⁷ within a questions environment, of a multi-question quiz.

¹⁵See the Exerquiz documentation.

¹⁶Between questions started by the `\item` command, and following the `\end{solution}` of the previous item, if there is a previous item, if there is a solution.

¹⁷Between questions started by the `\item` command, and following the `\end{solution}` of the previous item, if there is a previous item, if there is a solution.

References

- [1] Acrobat JavaScript Scripting Reference Version 8.0., Adobe Systems, Inc., 2007
http://www.adobe.com/go/acrobat_developer
50, 51, 53, 54, 63
- [2] Acrobat JavaScript Scripting Guide, Version 8.0., Adobe Systems, Inc., 2007
http://www.adobe.com/go/acrobat_developer
- [3] pdfmark Reference Manual, Version 8.0, Adobe Systems, Inc., 2007
http://www.adobe.com/go/acrobat_developer
58
- [4] PDF Reference, Version 1.7., Adobe Systems, Inc., 2007
http://www.adobe.com/go/acrobat_developer
8, 52