

talkpal

A $\text{\LaTeX}$ -based presentation software created by

Palash Baran Pal

What the package can do

The package `talkpal` is a $\text{\LaTeX}$ -based style file which allows you to create a pdf file with the following characteristics:

1. Using a multi-media projector, you can project your talk on the screen in different slides.
2. The slides can be divided into different segments which can appear one at a time, at the push of one key on the keyboard or one click of the mouse.
3. You can also use hyperlinks to flash a different portion of the file, and come back to the original place.

The package

The distribution contains the following files:

manual.tex

manual.pdf

talkpal.sty

slidekey.pl

The **.sty** file is the style file that will be needed for preparing presentations. The **.pdf** file is the one you are looking at now, which helps you understand how the **.sty** file works. The **.tex** was used to create the **.pdf** file, and you will not need it unless you want to see the input commands. The **.pl** file is a keyboard shortcut explained in the homepage of [talkpal](#).

Since you are reading a PDF file in a PDF viewer software, I will assume that:

- You know how to navigate in a PDF file. For example, you know how to go to the next page or the previous page in a file.
- You know how to use the full screen view. This is essential for giving presentations produced by this software. For viewing this file privately, it is not essential except in places where hyperlinks are described and used.

Notation used in this manual

Here are some notations used in writing this file.

1. The key (or keys, from the keyboard or from the mouse) used for going to the next page will be denoted by .
2. Names of pre-existing files and packages appear in **this color**.
3. Commands defined in standard L^AT_EX or in pre-existing packages will appear in **this color**. So will bash commands like **latex**.
4. Special **talkpal** commands will appear in **this color**.
5. Any variable name will appear *in this font and color*. You need to put an acceptable name there. For example, if you see *colorname*, you might put red or blue, but not Newton.

Okay, now use  to go to the next slide.

talkpal vis-à-vis other similar softwares

I have not used all presentation softwares mentioned here, so my comments are based on what I heard from others.

Like other softwares with similar objective, talkpal can show the text in slides and segments, using keyboard buttons to move from one to another.

It cannot use animation like **Power Point** does. On the other hand, you can use the rich formatting features of $\text{\LaTeX}$, including equations and tables etc. Other $\text{\LaTeX}$ -based programs like **beamer** are presumably more elaborate, but they are also much more complicated. They have to use separate document classes, so that input files are very different from ordinary $\text{\LaTeX}$ files. On the other hand, talkpal, based on the **article** class and a style file whose size is less than 3 kb, can do most of the things that are necessary for giving a talk.

Some time ago, I created another presentation software called **pptalk**. The present **talk_{pal}** is very similar to that one. The main difference is that **talk_{pal}** uses **pdflatex** to produce a **pdf** file directly, whereas the earlier program used **latex** followed by **dvips** to produce a **ps** file which could then be converted to a pdf file. The pros and cons of these two options need not be discussed here. Since more people use **pdflatex** these days, it seems that **talk_{pal}** will be more useful.

How to use `talkpal`

First, you need to create the `LaTeX` file following the instructions given later in this document.

Suppose the name of this `.tex` file is `xyz.tex`. Issue the command

`pdflatex xyz`

from your terminal or the pull-down menu on your editor. This produces a file called `xyz.pdf`. Open it on your computer screen using any standard pdf reader and project it by using the hardware available for that.

How to use `talkpal`

First, you need to create the `LaTeX` file following the instructions given later in this document.

Suppose the name of this `.tex` file is `xyz.tex`. Issue the command

`pdflatex xyz`

from your terminal or the pull-down menu on your editor. This produces a file called `xyz.pdf`. Open it on your computer screen using any standard pdf reader and project it by using the hardware available for that.

The rest of this manual describes how to create the input `.tex` file that is necessary for the procedure.

How to organize the file

You don't need to use a new class file in order to write your talk with `talkpal`. You can use the `article` class file. Use the package `talkpal`. In other words, start your file as follows:

```
\documentclass[12pt]{article}  
\usepackage{talkpal}
```

and of course, with the commands for beginning and ending the document. In the middle, you can write L^AT_EX in the usual way, i.e., as one does for writing a paper. On running `\pdflatex`, you will then obtain a file which will be presentable. In this case, `talkpal` merely fixes the size of the pages. You can force pagebreaks at suitable points by using the `\newpage` command.

However, with `talkpal`, you can do much more, as described below.

A few pieces of recommendation:

- Be careful to include the **12pt** option in the **\documentclass** declaration, as shown above. The sizes of fonts have been set with this option in such a way that appears nice on the projected screen.
- Do not use any of the page style commands like **\textheight** or **\textwidth**. These have already been set by **talkpal** to some optimum values which will suit a screen presentation.
- You may include other packages. But put them before the **talkpal** package so that nothing in **talkpal** is overridden by them.

Some “don’ts”

- Do not include the following packages in your .tex file, because they are already included in `talkpal`:

color	graphicx	lastpage
newcent	fancyhdr	hyperref

- Do not get turned off by the use of colors in this manual. Some of them, like the main font color, are defaults, but you can change them by commands shown later. Some others are defined within this document and have nothing to do with the style file for `talkpal`, like the colors in section headings and in displaying different commands. You can check the file **manual.tex** to find what has been defined there.

Dividing text into slides

You need to decide how much (or how little) of text you want to put in a particular slide. When you want to begin the slide, type

`\beginslide`

Then continue with whatever you want to put there, typing things just as in a normal $\text{\LaTeX}$ file.

Divide things into paragraphs as you want, put text and equations as necessary. When you finish typing the contents of the slide, type

`\endslide`

After that, use the **`\beginslide`** command to start a new slide.

If you put too much in a slide, text will spill at the bottom of the slide. You need to reconsider where to end the slide.

Dividing slides into segments

For a presentation, it is best to unfold the contents of a page in a number of segments, one segment appearing at a time. This way, the members of the audience are not exposed to a flurry of material at a time.

For this, start writing the slide with the first segment, after the `\beginslide` command. When you want to start the next segment, just type `\newsegment`

and then write the text for the next segment. Continue this process, adding more segments if you want, until you reach the end of the slide with `\endslide`.

Let us see an example. Notice that there is a huge empty portion at the bottom of this slide. This is because there are more segments down there which are hidden. Use  now.

Let us see an example. Notice that there is a huge empty portion at the bottom of this slide. This is because there are more segments down there which are hidden. Use  now.

Now you see the next segment! You can continue this way, introducing more segments. In principle, the number of segments on a page can be unlimited. However, the height of the page and the size of the font introduces some obvious restrictions.

Want to see more on this page? Hit .

Let us see an example. Notice that there is a huge empty portion at the bottom of this slide. This is because there are more segments down there which are hidden. Use  now.

Now you see the next segment! You can continue this way, introducing more segments. In principle, the number of segments on a page can be unlimited. However, the height of the page and the size of the font introduces some obvious restrictions.

Want to see more on this page? Hit .

From now on, hit  whenever you want to see some more material on the screen: whether material continued on the same slide or on a different slide.

You can put the command `\newsegment` almost anywhere in the file. You can put it at the end of any paragraph, as we have done once or twice earlier and will do more often from now on.

You can put the command `\newsegment` almost anywhere in the file. You can put it at the end of any paragraph, as we have done once or twice earlier and will do more often from now on.

But the command can also be put in the middle of a paragraph, as you will find out if you hit  now.

You can put the command `\newsegment` almost anywhere in the file. You can put it at the end of any paragraph, as we have done once or twice earlier and will do more often from now on.

But the command can also be put in the middle of a paragraph, as you will find out if you hit  now. You can put the command before or after an equation, before or after a table, or even in the middle of a word like (let's see if I can spell it right) “super

You can put the command `\newsegment` almost anywhere in the file. You can put it at the end of any paragraph, as we have done once or twice earlier and will do more often from now on.

But the command can also be put in the middle of a paragraph, as you will find out if you hit  now. You can put the command before or after an equation, before or after a table, or even in the middle of a word like (let's see if I can spell it right) “supercalli

You can put the command `\newsegment` almost anywhere in the file. You can put it at the end of any paragraph, as we have done once or twice earlier and will do more often from now on.

But the command can also be put in the middle of a paragraph, as you will find out if you hit  now. You can put the command before or after an equation, before or after a table, or even in the middle of a word like (let's see if I can spell it right) “supercallifragi

You can put the command `\newsegment` almost anywhere in the file. You can put it at the end of any paragraph, as we have done once or twice earlier and will do more often from now on.

But the command can also be put in the middle of a paragraph, as you will find out if you hit  now. You can put the command before or after an equation, before or after a table, or even in the middle of a word like (let's see if I can spell it right) “supercallifragilistic

You can put the command `\newsegment` almost anywhere in the file. You can put it at the end of any paragraph, as we have done once or twice earlier and will do more often from now on.

But the command can also be put in the middle of a paragraph, as you will find out if you hit  now. You can put the command before or after an equation, before or after a table, or even in the middle of a word like (let's see if I can spell it right) “supercallifragilisticexpi

You can put the command `\newsegment` almost anywhere in the file. You can put it at the end of any paragraph, as we have done once or twice earlier and will do more often from now on.

But the command can also be put in the middle of a paragraph, as you will find out if you hit  now. You can put the command before or after an equation, before or after a table, or even in the middle of a word like (let's see if I can spell it right) “supercallifragilisticexpiali

You can put the command `\newsegment` almost anywhere in the file. You can put it at the end of any paragraph, as we have done once or twice earlier and will do more often from now on.

But the command can also be put in the middle of a paragraph, as you will find out if you hit  now. You can put the command before or after an equation, before or after a table, or even in the middle of a word like (let's see if I can spell it right) “supercallifragilisticexpiallidocious”.

You can also put the command within groups. By **group**, I mean anything which is between a matching **\begin** and **\end** pair, or a **\left-****\right** pair in math mode. It can be a table, an environment like quote or enumerate, an equation, part of math mode which uses matching brackets.

You can also put the command within groups. By **group**, I mean anything which is between a matching **\begin** and **\end** pair, or a **\left-\right** pair in math mode. It can be a table, an environment like quote or enumerate, an equation, part of math mode which uses matching brackets.

However, in this case you will have to do a bit more. The command **\newsegment** works only within a group. So, when the group ends, if you want to continue with the same segment, put the command

\samesegment
and keep writing.

For example, in writing

$$\chi =$$

For example, in writing

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

I had to put the `\samesegment` command at the end of the equation.

For example, in writing

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

I had to put the `\samegment` command at the end of the equation. And when I wrote

$$\mathbf{x} = \left[\frac{1}{a} + \right.$$

For example, in writing

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

I had to put the `\samesegment` command at the end of the equation. And when I wrote

$$x = \left[\frac{1}{a} + \frac{1}{b} \right] (y + 2)$$

using `\left[\cdots\right]`, I had to put the `\samesegment` command after the closing square bracket.

Vertical segmentation

The `\newsegment` command allows you to make horizontal segments. If you want vertical segments, you should use the `minipage` environment for that. See an example [here](#):

Vertical segmentation

The `\newsegment` command allows you to make horizontal segments. If you want vertical segments, you should use the `minipage` environment for that. See an example here:

This is the first segment,
which can contain anything you like. Hit  to see the next segment.

Vertical segmentation

The `\newsegment` command allows you to make horizontal segments. If you want vertical segments, you should use the `minipage` environment for that. See an example here:

This is the first segment, which can contain anything you like. Hit  to see the next segment.

This is the second segment. Again hit  to see the third and last segment.

Vertical segmentation

The `\newsegment` command allows you to make horizontal segments. If you want vertical segments, you should use the `minipage` environment for that. See an example here:

This is the first segment, which can contain anything you like. Hit  to see the next segment.

This is the second segment. Again hit  to see the third and last segment.

Using too many vertical segments will make the segments narrow, which would not look good.

Vertical segmentation

The `\newsegment` command allows you to make horizontal segments. If you want vertical segments, you should use the `minipage` environment for that. See an example here:

This is the first segment, which can contain anything you like. Hit  to see the next segment.

This is the second segment. Again hit  to see the third and last segment.

Using too many vertical segments will make the segments narrow, which would not look good.

To see how this has been produced, hit  to go to the next slide.

The right panel shows the commands which produced the vertical segmentation in the previous slide. Note that one cannot leave blank lines between minipages, because that would result in the minipages occurring in different paragraphs.

```
\begin{center}
\begin{minipage}{0.3\textwidth}
This is the first...
\end{minipage}
%%
\hfill\newsegment
%%
\begin{minipage}{0.3\textwidth}
This is the second...
\end{minipage}
%%
\hfill\newsegment
%%
\begin{minipage}{0.3\textwidth}
Using too many...
\end{minipage}
\end{center}
```

How to use colors

You have already seen colors being used in this sample file. This has been done by using the package **color**. `talkpal` itself has this package as an input, so you should not introduce it explicitly when you are using `talkpal`.

How to use colors

You have already seen colors being used in this sample file. This has been done by using the package **color**. `talkpal` itself has this package as an input, so you should not introduce it explicitly when you are using `talkpal`.

The **color** package defines two commands for implementing colored text. One of them is `\textcolor`, used for a local change of color. The syntax is

`\textcolor{colorname}{Text material}`

which makes the “*Text material*” to appear in the color given as “*colorname*”. This command has been suitably redefined in `talkpal`, and will work exactly the same way within `talkpal`.

On the other hand, you can use the command

`\color{colorname}`

which is a declaration valid for the rest of the document.

If you use it at the beginning of your document or between two slides (i.e., after an `\endslide` and before the next `\beginslide`), the change will affect the rest of the document. If you use it within a slide (as it has been done at the beginning of this sentence), it will work only until the end of the slide. You will see the normal `textcolor` returning when you go to the next slide.

Color names

The basic color names appear in the file **color.sty**, which is part of the color package. They are: red, blue, green, cyan, magenta, yellow, and the non-colors white and black.

Color names

The basic color names appear in the file **color.sty**, which is part of the color package. They are: red, blue, green, cyan, magenta, yellow, and the non-colors white and black.

You can also create new colors. For example, a color called ***RONG*** can be defined by any of the following commands, where each p_i is a number between 0 and 1:

```
\definecolor{RONG}{rgb}{ $p_1$ ,  $p_2$ ,  $p_3$ }  
\definecolor{RONG}{cmyk}{ $p_1$ ,  $p_2$ ,  $p_3$ ,  $p_4$ }  
\definecolor{RONG}{gray}{ $p_1$ }
```

The last command produces only shades of gray. Try these out.

Background color

You have probably noticed that the background color of the slides changed after Slide 8.

It changed to a dark blue, because the command

```
\setbasecolor{darkblue}
```

was issued before the `\beginslide` corresponding to that slide. The color `darkblue` has been defined by the command

```
\definecolor{darkblue}{rgb}{0,0,0.5}
```

as explained earlier on Slide 22. This change is not part of the `talkpal` style file. It has been used in this manual file, to give examples of how you can change colors. Look out for the next change in the base color.

Suggestions on colors

I have a few recommendations on the use of colors. None of these is binding.

1. Use your judgement for color contrasts. Needless to say, the contents of a slide will not be very legible if you use orange letters on a yellow background.
2. Use dark backgrounds and light-colored text for best results. Light-colored backgrounds put unnecessary and bothersome glare on the eyes of the people in the audience.
3. It is best not to define a new color unless you absolutely need to do so. But if you need to, there is no restriction.
4. Background colors may be changed while moving from one section of the talk to the next. That's what I have done in this document.

Math mode

You can use math mode in much the same way that you use it for a usual $\text{\LaTeX}$ document, keeping the following recommendations in mind:

1. Normal math mode font for $\text{\LaTeX}$ is too thin, and does not look very good on slides. I advise using `\usepackage{euler}` in the preamble of your file. I have used it for producing this document.
2. Do NOT cross-refer equations by putting labels if you use multiple segments in a slide. I am against labeling and cross-referencing anywhere, because it is useless to say something like “...as shown in Eq. (17) earlier...” during a talk. So, for displayed equations, use either the `eqnarray*` environment, or use delimiters `$$` at both ends.

Displaying figures

Figures are included by the command `\includegraphics`. The file `graphicx.sty` is included in `talkpal.sty`, so you should not ask your `.tex` file to include it again. If the `\includegraphics` command does not appear in the first segment of a slide, you need to hide your picture until the proper segment arrives. For this, you need a slightly modified command:

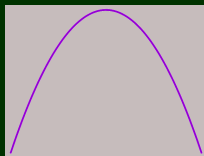
```
\putfigure{pictureheight}{other size commands}{filename}
```

The middle part can be empty. But the declaration of the height is essential, for which the amount should be specified without `\height=`. Examples:

```
\putfigure{0.3\textheight}{width=0.7\textwidth}{parabola.png}  
\putfigure{5cm}{}{parabola.png}
```

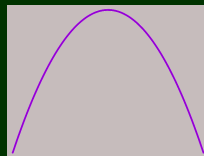
Do NOT put the figure in a **figure** environment, i.e., something starting with `\begin{figure}` and ending with `\end{figure}`. The **figure** environment turns the figure into a floating object, and L^AT_EX decides where to put it. Just for creating the figure, the `\includegraphics` is good enough, as modified here in `\putfigure`.

Do NOT put the figure in a **figure** environment, i.e., something starting with `\begin{figure}` and ending with `\end{figure}`. The **figure** environment turns the figure into a floating object, and L^AT_EX decides where to put it. Just for creating the figure, the `\includegraphics` is good enough, as modified here in `\putfigure`.



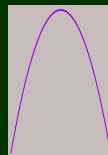
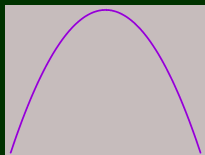
Here is an example of what `\putfigure` can do:

Do **NOT** put the figure in a **figure** environment, i.e., something starting with `\begin{figure}` and ending with `\end{figure}`. The **figure** environment turns the figure into a floating object, and L^AT_EX decides where to put it. Just for creating the figure, the `\includegraphics` is good enough, as modified here in `\putfigure`.



Here is an example of what `\putfigure` can do:

If you want to put multiple pictures side by side, use **minipage** environments, as discussed earlier in the section on “Vertical segmentation”.



Tables and other floats

For reasons mentioned in connection with figures, do not use the **table** environment that $\text{\LaTeX}$ provides. The same applies to all environments that put some stuff as float.

Tables and other floats

For reasons mentioned in connection with figures, do not use the **table** environment that $\text{\LaTeX}$ provides. The same applies to all environments that put some stuff as float.

A	B	C
D	E	F
G	H	I

Just for creating a table (an example given above), the **tabular** environment is good enough. If the entries contain too many math symbols, you can also use the **array** environment, putting the table within an equation.

Tables and other floats

For reasons mentioned in connection with figures, do not use the **table** environment that L^AT_EX provides. The same applies to all environments that put some stuff as float.

A	B	C
D	E	F
G	H	I

Just for creating a table (an example given above), the **tabular** environment is good enough. If the entries contain too many math symbols, you can also use the **array** environment, putting the table within an equation.

Do not put the **\newsegment** command within a table.

Using **hyperref**

As said before, cross-referencing is a bad idea. If you must want to refer to something mentioned elsewhere in the document, use **hyperref** commands. This package is also included in **talkpal**, so you must not include it again.

Using **hyperref**

As said before, cross-referencing is a bad idea. If you must want to refer to something mentioned elsewhere in the document, use **hyperref** commands. This package is also included in **talkpal**, so you must not include it again.

Hyperref commands appear in pairs. At the source, use the command

`\hyperpair{AAA}{BBB}`

where the variables are labels of your choice. At the target, put the same command, with the same labels in opposite order, i.e.,

`\hyperpair{BBB}{AAA}`

This way, you will get two arrows, one at the source and one at the target. If you click on them, you can go back and forth.

Using **hyperref**

As said before, cross-referencing is a bad idea. If you must want to refer to something mentioned elsewhere in the document, use **hyperref** commands. This package is also included in **talkpal**, so you must not include it again.

Hyperref commands appear in pairs. At the source, use the command

`\hyperpair{AAA}{BBB}`

where the variables are labels of your choice. At the target, put the same command, with the same labels in opposite order, i.e.,

`\hyperpair{BBB}{AAA}`

This way, you will get two arrows, one at the source and one at the target. If you click on them, you can go back and forth. Check it by pressing on this arrow  to go to the last page, and press the arrow there to come back here.

You may also put an optional argument in the command `\hyperpair`, which will appear as the symbol for the hyperref spot. This argument can be different for the source and the target, if you wish. Here is an example of how to do it:

`\hyperpair[\spadesuit]{BBB}{AAA}`

Try using it for going to a certain place near the end of this document. a

You may also put an optional argument in the command `\hyperpair`, which will appear as the symbol for the hyperref spot. This argument can be different for the source and the target, if you wish. Here is an example of how to do it:

`\hyperpair[\spadesuit]{BBB}{AAA}`

Try using it for going to a certain place near the end of this document. ♠a

Warning: Hyperlinks do not work properly unless the full-screen view is used.

Flashing

You might also want to flash a picture or a big formula, but do not want to carry it in the rest of your slide. This can be done by putting the object in a picture file such as **flashpic.png**, and then issuing a command like this:

```
\flash{0.2} {\includegraphics[height=0.8\textheight]{flashpic.png}}
```

The first argument raises the flashed object by that fraction of **\textheight**. The second argument is the object to be flashed.

Flashing

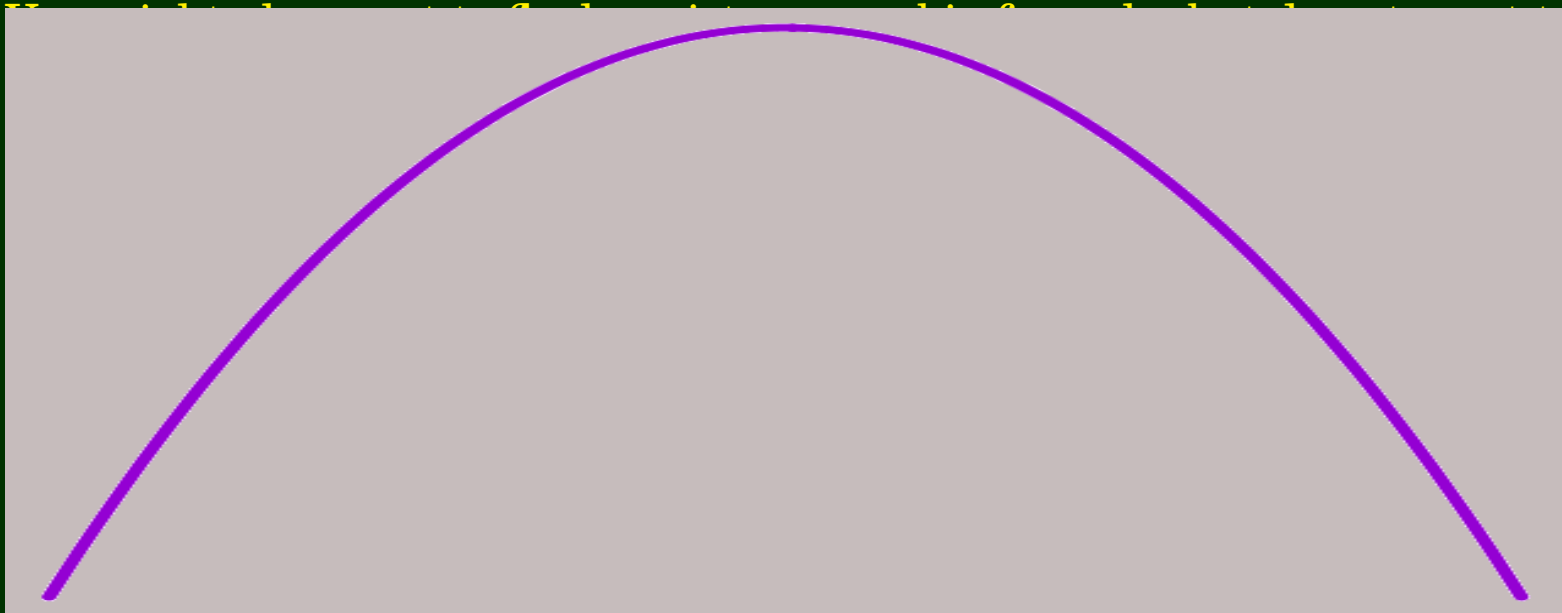
You might also want to flash a picture or a big formula, but do not want to carry it in the rest of your slide. This can be done by putting the object in a picture file such as **flashpic.png**, and then issuing a command like this:

```
\flash{0.2} {\includegraphics[height=0.8\textheight]{flashpic.png}}
```

The first argument raises the flashed object by that fraction of **\textheight**. The second argument is the object to be flashed.

Try hitting  here to see such a flash appear.

Flashing



to carry it
the file
}}
nt. The

Flashing

You might also want to flash a picture or a big formula, but do not want to carry it in the rest of your slide. This can be done by putting the object in a picture file such as **flashpic.png**, and then issuing a command like this:

```
\flash{0.2} {\includegraphics[height=0.8\textheight]{flashpic.png}}
```

The first argument raises the flashed object by that fraction of **\textheight**. The second argument is the object to be flashed.

Try hitting  here to see such a flash appear.

As you go to the next segment by hitting , the “flash”ed component disappears.

On defaults, and how to change them

The default font family for L^AT_EX is **cmr**. These fonts are not suitable for slide presentations. So I have reset the default to **newcent**. The **palatino** and the **times** families also work quite well. If you want to use **palatino**, for example, put the command

```
\usepackage{palatino}
```

For **helvetica**, I don't know whether there is a similar package, but you can use the command

```
\renewcommand\familydefault{\sfdefault}
```

Such commands must be put in the preamble of the file after the **talkpal** package has been included.

Other defaults are summarized in the following table, with examples of how to change them.

Parameter	Default value	Example of change
Font size	Large	<code>\def\defaultsize{large}</code>
Text color	yellow	<code>\color{white}</code>
Background color	black	<code>\setbasecolor{blue}</code>
Foottext	Created with talkpal	<code>\def\foottext{<i>My talk</i>}</code>
Footcolor	white	<code>\def\footcolor{pink}</code>

Please note that some of the commands need a leading `\def`, while others do not.

Summary of commands

Here is a summary of the special commands for `talkpal`.

- `\beginslide ... \endslide` Denotes the beginning and end of a slide. The contents of the slide goes in the middle, in place of the three dots.
- `\newsegment` Denotes the beginning of a new segment in a slide. Not required for the first segment in a slide, which begins with the `\beginslide` declaration.
- `\samesegment` If the `\newsegment` appears within a group, this command has to be used at the end of the group to reassert the continuation of the segment after the group. a

\putfigure Necessary for placing figures inside the presentation.

\flash Flashes a picture in a particular segment, which vanishes when the next segment is activated. The “picture” can be anything in a picture file, even a long formula.

\setbasecolor Changes the background color of slides.

\def\defaultsize Defines the default size of text.

\def\foottext Changes the foottext.

\def\footcolor Changes the color of the material in the footer.

As an example, we change the foottext and footcolor after this slide, with appropriate commands after the **\endslide** of this page.

It is also useful to master the use of some commands which are not `talkpal` specials, but are not used while writing papers and books.

`\definecolor` Used for defining colors.

`\textcolor` Redefines the text color locally.

`\color` Defines the text color either till the end of the slide or till the end of the document, as explained earlier.

`\pageref{Lastpage}` Puts the number of the last slide. For example, footers in this document have been produced by the command

```
\foottext{Created with talkpal \hfill Slide \arabic{page} of  
\pageref{LastPage}}
```

These commands are defined in the standard packages `color` and `lastpage`, both of which are included in `talkpal`.