User's Guide to the Free Speech Calculus Text Version 0.5

W. Ethan Duckworth

August, 2005

Contents

1 Introduction

How would you like to have a Calculus text that you could customize to the way you like to teach Calculus? How would you like to have a text where you can print out just the examples? Or just the definitions? Or suppress all the examples? How would you like it if there could be an electronic version with hyperlinks to definitions, hyperlinks for extra detail, hyperlinks to historical information? How would you like it if this text was free in all the usual senses: free as in speech (you can use it how you like, other people can use it how they like) free as in open to change (you can add examples, pictures, historical interludes to your version or submit them to the master version) free as in ice cream (you can give it away for no charge to your students).

Well, I think the technology exists to make all these things and more possible. I do not claim that I have accomplished all of these goals yet. At present (August 2005) we have about 400 pages of of text (culled from a variety of sources), and graphics. This means that we need considerably more material to be complete. I have implemented LaTeX code which allows many options and combinations in how the document can be processed. I have not added all of the key-values (a type of meta-data for the material) for showing/hiding all combinations of examples, definitions, etc. I have not added any of the hyperlinks for the electronic version, though this is fairly easy to do with standard packages and using PDF format. (Note that the goal of widespread use of this text makes the PDF format the best choice for the final product.)

This calculus text is copyrighted under the Gnu Free Documentation License (GFDL). Loosely speaking here are the terms of this license:

- You are free to copy, redistribute, change, print, sell, and otherwise use in any manner part or all of this document,
- Any work derived from this text must also be covered by the GFDL. (This only applies to that part of your work derived from this text. It is up to you whether those parts of your work which are not based on this text is covered by the GFDL. Also, you can quote, with attribution and copyright notice and subject to fair use provisions, from this text like you would from any copyrighted work.)
- Anyone distributing works covered by the GFDL must provide source code or other editable files for the material which is distributed. In the case of this text that means the LaTeX code, as well as the source documents for creating the graphics.
- If you make changes to this text and redistribute them, you should name the finished product something different than "The Free Speech Calculus Text" or "The Free Speech Calculus Text: original version". You may

choose derivative names like "The Free Speech Calculus Text: the John Doe version".

The material is a combination of independent works by W. Ethan Duckworth, Paul Garrett, Michael Livshits, and the various contributors to Wikibooks. All the independent works were licensed under the GFDL; W. Ethan Duckworth combined them into one whole, added some material and LaTeX code (this is the type of thing that the GFDL encourages).

2 Who are the users?

If you think about it, an ordinary Calculus textbook has two types of users: students and teachers. If we oversimplify we might say that the students mostly look at the examples and the exercises. The teachers look at the ordering of sections, what's in each section, and the exercises.

The Free Speech Calculus Text will probably have four types of users.

The student. At present I think the student will only use material that the teacher prepares for him or her. Once the text is more complete, here is how I envision a student using it.

Firstly, the electronic version of the text can complement the printed version. In an electronic version there could be extra historical information, there can be keyword searches, hyperlinks to definitions, etc.

In the future, I also imagine having a graphical interface for a student to be able to choose to see only examples. Or to see all the examples using $\sin(x)$. After they make their choices, TeX would run in the background (invisibly) and they would have a sub-book. The graphical interface can actually be quite simple, because the configuration required can be answered with a small set of yes or no type questions. The answers to these questions would be written out in a configuration file, and then TeX would run, using this configuration.

The teacher (aka a serious user). At present, the material is not complete, so I think a teacher would only use this text for a few extra examples, or for a few sections, not for the whole class. Once this text is complete, here's how I envision a teacher using it.

In addition to the uses that teachers currently have for textbooks, I imagine a teacher using the Free Speech Calculus Text more interactively. If the teacher wants to include more history, they could just say "show me the history". If they want to omit trigonometry or make it come late in the class, they can choose to do this.

At present (more correctly, once the material in the text is complete), the teacher can do this with a small amount of knowledge of LaTeX. In the future, I imagine various versions of the text that the typical teacher could choose from, even with zero knowledge of LaTeX. Farther in the future I again imagine a graphical interface, as mentioned for the student user.

Contributors. At present the text are still incomplete, and I welcome contributions. By all means, contribute any way you like. Take the text and teach a class with it. Or extract some examples from them. Hand out supplements, etc. Probably if you try to use the text you will see something you wish was there, but isn't. This is the perfect thing to contribute because you're interested in it, and because there's probably someone else who wishes it was there too. If you do any of these things and want to return something to the community, share what you've added, either by posting it on the web, or by contacting me and having it folded back into the main distribution of the Free Speech Calculus Text. You are also welcome to submit even small changes, like grammatical corrections, or inserting a sentence.

At present, I imagine these text as being like a moderated Wiki. By this I mean that virtually any changes that are submitted to me will be incorporated, and that I hope you submit complete changes, not just suggestions. In fact, in the future, we might turn the text into some sort of Wiki, or Wiki-hybrid; thus allowing anyone to make changes directly without any intervention on my part.

A consequence of this policy will be, I hope, a rapidly growing text with a large number of alternative approaches and extra material. (If you feel that I accept too many/few contributions then, because the text is licensed under the GFDL, you are welcome to copy the whole thing and create an alternate project with different rules, but of course with the same license.) For this reason, one of the main goals in writing the LaTeX code has been to make it easy to show/hide parts of the text. In the future, I imagine there being more material than anyone would want in a single book and, if this becomes the case, one of the things contributors could do is provide different versions which show different subsets of the material. So there could be a standard version, a friendly version, a formal version, a version with extra history, a version based on infinitesimals, a version that doesn't use limits, etc. The tools for doing this are described below.

Hackers. Here I of course mean people who like fooling around with computers and code, not malicious people who like to attack computers which belong to others. I describe below some goals I would have below for this type of user, though hackers are notoriously prone to doing what they like rather than following directions! Good.

3 How a teacher or serious user uses the text

This is the actual "user guide": what are the commands, how do they work, etc. I am not going to cover LaTeX basics, rather things that are particular to this text.

Figure 1: Simple `master.tex` file

```
\documentclass{article}
\usepackage{free_calc}
\begin{document}
\chapter{Derivatives}
  \include{rules}
  \include{derivative_applications}
\chapter{Antiderivatives}
  \include{rules}
  \include{integral_applicaitions}
\end{document}
```

3.1 Inputting files

The file `free_calc.sty` loads various packages and defines many commands for use in the text. The file is fairly well commented, so if you're a hacker, it might contain everything you need to understand what's going on.

All of the text files, graphics files, etc. are read into some main file. I call the main file `master.tex`. In the future there might be different versions of this file, each producing a different version of this text, as described in the paragraph above for contributors. The files are called in with `\include`, and can thus be turned off with `\includeonly`, though `\includeonly` is best used for editing, not for producing a finished alternate version.

For example, suppose `master.tex` looked like figure 1. Then four files are included. If we added a command `\includeonly{rules}` in the preamble, then only one file, `rules` will be included. Similarly, one can comment out using %, any file that one doesn't want to include. The beginning of the real `master.tex` file looks a lot more like the example shown in figure 2. As you can see from that figure, I have chosen to make "fine-grained" sub-files; this allows the greatest flexibility in turning them on and off and in rearranging them. The cost is of course having to keep track of a couple hundred files. (Again, if you don't like having so many files, you're welcome to take this collection, merge them into one file, and distribute your version on the web.)

Already one can see that a huge number of Calculus texts could be created by manipulating the included text files. However, the Free Speech Calculus Text is far more flexible than this.

3.2 Pseudo-paragraphs

For these notes to have greater flexibility to hide/show material than would be possible by using `\includeonly` and % on text files, the material in the files must be chopped into many pieces. These pieces are defined by environments I

Figure 2: Beginning of a real `master.tex` file

```
\documentclass{book}
\usepackage{free_calc}
\title{\huge\bf The Free Speech Calculus Text}
\author{Various authors, Gnu Free Documentation License}
\date{November 22, 2004}

\begin{document}

\section*{Introduction}
  \include{text_files/introduction_to_whole}

\chapter{Background}
  \include{text_files/introduction_to_background}

  \section{Positive, negative numbers and inequalities}
    \include{text_files/basics_about_numbers}
    \include{text_files/inequalities}
    \include{text_files/rules_of_basic_algebra}
    \include{text_files/interval_notation}
    \begin{homework}
      \include{exercises/inequalities}
    \end{homework}

  \section{Functions}
    \include{text_files/what_is_a_function}
    \include{text_files/list_of_basic_functions}
    \include{text_files/function_notation}
    \include{text_files/function_examples}
    \include{text_files/combining_functions}
    \include{text_files/domain_and_range}
    \include{text_files/one_to_one_functions_and_inverses}
    \include{text_files/mathematical_models}
    \begin{homework}
      \include{exercises/function_domains}
      \include{exercises/functions_exercises}
    \end{homework}

  \section{Analytic geometry}
    \include{text_files/basics_of_graphs}
    \include{text_files/lines_and_circles}
    \include{text_files/graphing_functions_without_calculus}
    \begin{homework}
      \include{exercises/lines_and_circles}
    \end{homework}

  \section{End of chapter problems}
    \begin{homework}
      \include{exercises/funny_problems}
    \end{homework}


\chapter{Limits}
  \section{Elementary limits}
```

call pseudo-paragraphs. Here's the list of pseudo-paragraphs at present:

`axiom`	`notation`
`comment`	`procedure`
`cor` (prints **Corollary**)	`program`
`definition`	`proof`
`derivation`	`rule` (prints **Rule**)
`discussion`	`strategy`
`example`	`theorem`
`fact`	`test`
`lemma`	

It's easy to add new pseudo-paragraphs, although I'll wait to explain how until later (in the subsection on how a contributor uses the text).

All the material in the text is contained in pseudo-paragraphs. Each pseudo-paragraph has a list of key-values as an argument. It is the key-values which control which pseudo-paragraphs are displayed, and which hide their contents. For example, we might type something like

```
\begin{example}{author=duckworth,author=newton,uses=sin}
Let's show that the derivative of $\sin(x)$ is $\cos(x)$.
Etc.
\end{example}
```

Then this pseudo-paragraph would be controlled by the key-values "author=duckworth", "author=newton", "uses=sin".

3.3 Controlling pseudo-paragraphs with `\Show`

For the user to control which pseudo-paragraphs are shown and which are hidden, one uses the `\Show` command. The argument to `\Show` consists of a boolean statement which is built from the standard logical constructors `or`, `not`, `and`, `(`, `)`, and from the pseudo-paragraph names and key-values. Each pseudo-paragraph which follows that `\Show` statement will be displayed if and only if it satisfies the boolean statement defined by `\Show`. A `\Show` statement can be entered anywhere in the document `master.tex` and it will affect the pseudo-paragraphs which follow it. More than one `\Show` command can be entered; each time it is entered the previous command is replaced entirely. Because I am either not clever enough or do not have enough time, the items in the argument to `\Show` must be separated by commas, which looks a little funny.

3.3.1 Examples of the Show command

For example, suppose one enters, the following `\Show` command in `master.tex`.

```
\Show{(,author=duckworth, and, author = newton,),or,uses=sin}
```

Figure 3: Examples of pseudo-paragraphs

```
% 1
\begin{example}{author=duckworth,uses=limits}
  Show that the limit of .... Etc.
\end{example}

%2
\begin{example}{author = newton, uses =sin}
Show that the derivative of $\sin(x)$ is \cos(x).
\end{example}

%3
\begin{lemma}
{author = wikibooks, author=duckworth, uses=e^x,uses=limits,style=formal}
The derivative of $e^x$ is $e^x$.
\end{lemma}

%4
\begin{discussion}{author=duckworth,author=newton}
Let's look at what the previosu lemma means, etc.
\end{discussion}
```

Then each pseudo-paragraph will be shown if and only if it makes this boolean statement true (a slightly more technical description of how this works is given below). For the list of environments in figure 3 the ones which will be shown are #2 and #4. If one entered

```
\Show{not,author=duckworth}
```

then only #2 would show. Note that in this case any key-value of the form `author = newton` is simply ignored.

In addition to key-values, names of pseudo-paragraphs may be given in a `\Show` command. Thus, if one entered

```
\Show{example,or,author=newton}
```

then environments #1,2 and 4 from figure 3 would be shown.

The command `\Show{}` removes all restrictions and shows everything.

The `\Show` command can be entered many times in a document. A simplified example is given in figure 4. In this example, the only derivative rules which would be shown are those which do use $\sin(x)$, $\cos(x)$, $\tan(x)$ but which do not use limits. All of the applications of derivatives would be shown, etc.

There are times when one wants to show almost all environments, but if a certain key type is present to only show particular values of that key. For example, the key type `style` has been defined and is meant to have values of the

Figure 4: Modified simple `master.tex` file

```
\documentclass{article}
\usepackage{free_calc}
\begin{document}
\chapter{Derivatives}
\Show{(,uses=sin,or,uses=cos,or,uses=tan,),and,not,uses=limits}
  \include{rules}
\Show{}
  \include{derivative_applications}
\chapter{Antiderivatives}
\Show{examples}
  \include{rules}
\Show{uses=volume_by_rotation}
  \include{integral_applicaitions}
\end{document}
```

form `style = formal`, `style = informal`, `style = standard`, or something like this. Not all environments have anything defined for `style`. Now suppose you wanted to write a "standard" text, and thus not show things that are too "formal" or "informal". You cannot issue a command of the form `\Show{style=standard}` because most environments have no style defined, and thus most environments will not pass this test. Instead, the way to enter this is

```
\Show{not,(,style=informal,or,style=formal,)}
```

The syntax for such a command is not very good because it requires the user to enter a list of all the alternatives to `style = standard`. In the future, it would be desirable to have a variation on the arguments to the `\Show` command so that `\Show{foo=empty, or, foo=bar}` where `empty` would be a technical value meaning "not present". Thus this command would mean something like "Show if the key-type `foo` is not present or if it is present and it equals `bar`".

3.3.2 Caveat: the Show command requires keys

There's one caveat to all these uses of `\Show`: they require key-values to be entered in the pseudo-paragraphs. At present this has been done for "author" throughout the text. The first couple of chapters also have the "label" and "uses" keys installed. In any but the functionality has been created to manipulate the keys, as described above. There is also a macro for adding new keys.

Here's the list of current key-value pairs as applied to pseudo-paragraphs:

Figure 5: Examples of the `label` key

```
% 1
\begin{example}
{author=duckworth,uses=limits,label=example:limit_of_sin_over_x}
  Show that the limit of .... Etc.
\end{example}

%2
\begin{example}
{author = newton, uses =sin, label=example:deriv_of_sin_is_cos}
Show that the derivative of $\sin(x)$ is \cos(x).
\end{example}

%3
\begin{theorem}
{author=duckworth, label=theorem:taylors_theorem,version=1}
Let $f(x)$ be a function whose $n$th derivative exists.  Then the
power series defined by etc.
\end{theorem}

%4
\begin{theorem}
{author=duckworth, label=theorem:taylors_theorem,version=2}
Let $f(x)$ be any a function whose $n$th derivative exists.  Define
the polynomial etc.
\end{theorem}
```

```
author                          requires
uses                            label
style                           version
importance
establishes
```

Most of these are just descriptors which can be exist to provide a link between the `\Show` command and the pseudo-paragraphs. However the `label` executes certain actions within the pseudo-paragraph.

3.3.3 Fine-tuning with `label`

The key-values described so far, `author`, `uses`, `style`, etc. are rather broad. By themselves, one could not obtain perfect control over which pseudo-paragraphs are displayed. The `label` key overcomes this difficulty. In principal each pseudo-paragraph has an almost unique label key, as shown in figure 5.

Thus, in addition to the `\Show` commands described above, one could issue commands of the form

```
\Show{label=example:deriv_of_sin_is_cos}
```

This allows for complete fine-tuning of which pseudo-paragraphs are displayed.

In addition, the `label` key writes a standard LaTeX label which can later be referred to with `\ref`. Thus, using figure 5, one could later write "By Taylor's Theorem `\ref{theorem:taylors_theorem}`". This use requires (1) that most pseudo-paragraphs have a unique label but that (2) truly alternative version's of the same result get the same label, but different versions. This is so one can automatically use `\ref{theorem:taylors_theorem}`, even if this use comes in an entirely different chapter from the lines which display Taylor's theorem.

3.3.4 Other arguments to Show

In addition to the key-values just listed, `\Show` can take the names of pseudo-paragraphs (as mentioned above), a list of which is given below.

There are also two arguments which are for editing purposes. If the word `files` is given then each pseudo-paragraph which shows will display the name of the tex file which contains that pseudo-paragraph. If the word `keys` is given then each pseudo-paragraph which shows will display it's list of keyvalues. Thus

```
\Show{keys,files,author=duckworth,and,author=newton}
```

will cause environment #4 in figure 3 to show, and it will print `author = duckworth , author = newton`, as well as the name of the tex file which contained it.

3.3.5 A special psuedo-paragraph: homework

The `homework` environment is a special case of a psuedo-paragraph. This environment is used to start a homework section. You use it like so:

```
\begin{homework}
\include{exercises/problems_on_quotient_rule}
\include{exercises/problems_on_product_rule}
\end{homework}
```

This environment responds to the `\Show` command like the other psuedo-paragraphs.

If you want to edit a certain batch of homework problems, then you should use the `\includeonly` command. However, `homework` includes two additonal files that you can't see in source code just given: a `begin_hw` file and a `end_hw` file. These contain the commands which format the exercises and execute an `enumerate` list. Thus, if you want to use `\includeonly` with homework, you should enter something like:

```
\includeonly{exercises/begin_hw, %
       exercises/problems_on_quotient_rule, exercises/end_hw}
```

The reason the files `begin_hw` and `end_hw` were created were so that you can type `\includeonly{text_files/product_rule}` and automatically *not* have the `enumerate` environment executed by the `homework` environment.

4 How a Contributor uses the text

First of all, you are free to copy the entire set of notes to your computer and keep all your contributions to yourself. However, if you choose to contribute to the collection that I'm maintaining and sharing, that would be great.

As I mentioned above, there's lots of room for contributions: of text material, graphics, LaTeX etc. Again, I will accept additive contributions with little oversight. Because of the tools built in to this text for showing/hiding things, I think there is little reason to remove material, although at present there are redundancies and duplications which should be removed.

WARNING

Anything you contribute should be free from copyright infringement. The simplest way to make sure this is the case is to write it yourself. You should not assume that you can use anything from a book, from the web, or even from someone's "free notes" unless there is a very explicit statement about the copyright, placing the work under the GFDL or in the public domain.

Information available on `www.wikipedia.com` and `www.wikibooks.com` is covered by the GFDL and may be incorporated in these notes. In fact, a significant part of this material came from the wikibooks website as it existed in November of 2004.

So how should one contribute? The answer has two parts: what sort of guidelines should be used for writing the stuff up, and what sort of guidelines should be used for getting your stuff into the main body of the text.

4.1 Getting your stuff into the main distribution

At present we don't have any fancy system for contributors; this isn't hosted on `www.sourceforge.com`, and we don't have a CVS system for keeping track of versions and contributors. So, send your contributions to me. (If you have knowledge about running a CVS or about putting stuff on `sourceforge` and think these tools would help the Free Speech Calculus Text, then contact me and let's talk.)

If you do send me something, please send me very specific instructions for where it should go; include the filename, the line number, the environment it follows etc.

If you have a small amount of text, you can just paste it into an email to me. If you have a whole new pseudo-section send it to me as an email attachment. If you've edited a file so much that it would be simpler for me to just replace

my copy with the one you've made, that's ok too, just attach the whole file, and again give me clear instructions.

I prefer line-endings to be in the UNIX format, though this is easily changed by any decent text-editor. I do request (though not require) that you use hard-wrap instead of soft-wrap. In other words, set your text editor so that it enters a return when your text reaches the right hand edge of the window. If you send me material that has not been wrapped in this fashion, I will (1) re-wrap it myself, and (2) communicate with you directly to find out if you didn't understand what I was asking for or if you have some reason not to do it this way.

If you've edited `free_calc.sty` much (i.e. if you're improving my LaTeX code) then of course test your code as much as possible before sending it to me. I'll test it too at my end.

4.2 Figuring out where to edit

Some TeX set-ups have what's called "reverse synchronization". This means that when you click on the dvi/pdf output, it sends you to your editor at the place which produced the output you clicked on.

If you don't have this (or haven't bothered setting it up), I've included something to help you. The command `\Show` can take arguments of "keys" and/or "files". The effect of this command will apply to all the pseudo-paragraphs which come after it; thus, you would usually put it in the pre-amble, but this is up to you.

When executed, this will show information which will help you figure out where and what to edit. `\Show{keys}` will cause each pseudo-paragraph to show the key-values in the print-out. This way you can see, for example, that a proof is by duckworth, and you will know to hide duckworth, if you want to get rid of it.

`\Show{files}` will cause each pseudo-paragraph to show which text-file it is contained in. This way, you will know where to go to edit it.

4.3 What to write

Well, mostly I figure you should contribute anything you like. If you've got a favorite example, put it in. If you've got some notes that you've typed up, chop them into pseudo-paragraphs, and put them in. If you see something missing, put it in.

Here's some big things that I know are missing.

Key-values. As described above the `\Show` command relies upon key-values being entered for each pseudo-paragraph. At present all pseudo-paragraphs have values entered for `author`. The material in the Background and Limits chapters has been somewhat thoroughly processed, but the rest of the material has not been. In particular, each pseudo-paragraph should have an almost unique `label` key added. This creates a standard LaTeX label, as well as provides a key for fine-tuning the `\Show` command. These

labels should be somewhat lengthy and descriptive, rather than `label = ex1523`.

History. I think it would be nice to have short little bios for the major figures in Calculus. The more entertaining the better. Don't worry about them being too long, anyone who doesn't want them can turn it off!

Graphics. We need lots more graphics. It might be reasonable to aim for one graphic per page. Don't worry about having too many graphics, anyone who doesn't want so many can turn them off!

Exercises. We need lots of exercises. Either with or without solutions and answers. (See also the "hacker" goal below for implementing code for the homework.)

Alternative Calculus Implement a `master.tex` document to make alternative books. For instance, it would be nice to have `master_infinitesimal.tex` which makes the book take an infinitesimal approach to Calculus. There is also a "limits free" approach to Calculus, and it could have a `master_no_limits.tex` file.

4.4 Writing stuff up

I don't want to crimp anyone's style, but, for the sake of working on a common text, we should have some consistency. Unfortunately, "some consistency" means that it might take a little more work than if you were typing up your own test or notes, but I hope that "little" is accurate.

4.4.1 Pseudo-paragraphs

Everything you contribute should be in smallish pieces which I've called pseudo-paragraphs. At present most of the pseudo-paragraphs don't have extensive key-values, but it might be nice if you included at least the author, if it uses any important functions or techniques, and of course the `label` key.

I understand, and agree, that the use of pseudo-paragraphs gives the text an unfortunate chopped-up feel. This can be moderated later by turning off numbering and titles for some of the environments, like comments or discussion. Then these environments would blend in with the background. But I feel that using the pseudo-paragraphs is the best way to allow the mixing of different examples, theorems, etc, by different authors, and allow these things to shown/hidden.

4.4.2 Creating new pseudo-paragraph environments and new key-values

Although there's a pretty large list at present of pseudo-paragraph environments and key-values, there will undoubtedly be someone who wants to create new

ones. This makes sense, but use restraint as the whole package could become unwieldy if the lists become really large.

To create a new key-value called `foo` you would enter

```
\NewPPKey{foo}
```

After this has been done then any pseudo-paragraph could have entries like the following:

```
\begin{example}{author= duckworth, foo = bar}
Etc
\end{exmaple}
\end{verabatim}

To create a new psuedo-paragraph called {\tt foo} you would enter two
commands thus
\begin{verbatim}
\newtheorem{innerfoo}{\footitle}
\NewPP{foo}{Foo}{\thingy}
```

You have to create an environment called `innerfoo`; this doesn't have to be done with `\newtheorem`, but the `amsthm` package makes the `\newtheorem` powerful and convenient for our purposes. Thus, you can decide how the counter for `foo` should work by entering `\newtheorem{innerfoo}{\footitle}[section]` or `\newtheorem{innerfoo}[example]{\footitle}`.

The command `\footitle` is defined by the `\NewPP` command. For this to work you must use `\footitle`. The default for `\footitle` is the second argument to the `\NewPP` command, in this case **Foo**. You can re-define this as an optional argument when you use `foo`. Thus

```
\begin{foo}[Newton's Foo]{author = duckworth}
```

would make that particular `foo` environment have the title **Newton's Foo**.

The command `\thingy` defines an end of pseudo-paragraph symbol. For proofs, this is the `amsmath` command `\qed` which makes the box □. You can leave this symbol empty, as is the case for lemmas, theorems and corollaries. An illustration of a generic symbol to mark the end of examples, rules, definitions etc. is a horizontal line:

The command to create this line has been called `\linesep`. Here's the actual code I used to create the example environment:

```
\newtheorem{innerexample}{\textcolor{blue}{\exampletitle}}[section]
\NewPP{example}{Example}{\linesep}
```

The command `\textcolor{blue}{\exampletitle}` makes the Example appear in blue. The command `[section]` tells LaTeX to number examples within sections, so that the numbering restarts with a new section.

4.4.3 Pseudo-sections

Pseudo-sections are what I call the files that are included into `master.tex`. What should form a pseudo-section? A pseudo-section should be a cohesive part of a single section. Obviously, this is a little vague, or, if you like, a matter of preference. One way to figure out what should be a pseudo-section is if you can imagine someone wanting to hide/show/move the whole body of material. So, the quotient rule and the product rule could each form a pseudo-section; you would probably include them both in a single real section. Similarly, in the real section of integrating rational functions, you might include a pseudo-section for each of the following: polynomial division, completing the square, partial fractions, etc.

4.4.4 Notation

Of course we want the text to have somewhat consistent notation. Of course all Calculus books use more than one notation for derivatives, and we will follow this lead.

There are two ingredients in creating a moderate amount of consistency. (1) Don't be too idiosyncratic, or too technical, or too perfect etc. So, in Calculus I, we'll use $\frac{d}{dx}$ instead of D_x. We'll write $\int x^2\,dx$, not $\int x^2\,\mathtt{dx}$. We'll write out "for all x in $[0,1]$" instead of "$\forall x \in [0,1]$". (2) Use mark-up language for things that have specific meaning, especially things that have specific meaning that someone else might write differently. Thus use the command `\ddx` which means "the derivative with respect to x", instead of writing `\frac d{dx}`. Why the command? Well, $\frac{d}{dx}$ has specific meaning, it's not just a fraction with a d on top and d and x on the bottom. Plus, if we all use `\ddx` then all of our derivatives look the same. Moreover, if one of us is crazy enough to want their derivatives to look like D_x, $\mathbf{D_x}$, or $\frac{\mathrm{d}}{\mathrm{dx}}$ then all they have to do is re-define the command `\ddx`, and *presto*, all of the derivatives have changed their appearance.

It would be nice, but I doubt it'll happen, if we all used `\sin{x}` and `\arcsin{x}`. That way, by changing a single definition at the top, those who like to write $\sin x$ could have all their sins look this way and those who like to write $\sin(x)$ could have all their sins look this way. Similarly `\arcsin{x}` could be defined to produce $\sin^{-1}(x)$ or $\sin^{-1} x$, or $\arcsin x$, or $\arcsin(x)$ according to the teacher user's whim. I suspect that such notational enforcement will be too difficult to maintain.

Brows the file `free_calc.sty` for a list of built in notations, and feel free to add some.

4.5 Graphics

Of course LaTeX can import a huge variety of graphics. The poorest quality ones are bitmaps like JPEG, PNG or GIF. The nicest looking ones are in PostScript or PDF, or generated internally by packages like `eepic`.

4.5.1 Formats and openness

The fact that this text is licensed under the GFDL, that it will be used by and contributed to by a wide variety of people, and that it will have a wide variety of applications (i.e. text, electronic, web-based, etc), means that the format and approach to graphics might be different than what you do for your tests/lecture notes, etc. Here are further considerations of these points.

- The GFDL requires that any distribution of this text also provide source files for the text and the graphics. Thus, if you make a plot in `Maple`, there should be a `Maple` source file somewhere that creates that plot.
- Since many people might be looking at the source and modifying it, it would polite and nice to have that source be used by a program that is widely available, preferably on different platforms.
- Since many people might be looking at the source and modifying it, it would be nice and polite if that source is human-readable, not binary/cryptic XML. Thus, if you use `Maple` it is preferable to have the `Maple` code for a plot saved as plain-text, rather than as a worksheet.
- Since different uses might be made of this text, it is best to use tools which can export graphics into a wide variety of formats.
- If someone wants a particular application of this text that requires a particular graphic format, it would be nice if changing the graphics was reasonably easy. Since there will be hundreds of graphics, this means that the graphics should be done in such a way to allow batch-processing. In other words, there should be a small set of commands which would allow all the graphics to be re-generated. For `Maple` this is most easily done by creating a single source code file with the the plot-settings at the top, and all the plots afterwards. This has been done in the `plot_library.mpl` file within the `graphics_maple` directory. For `Xfig` all the fig files can be processed from the command line with the `fig2dev` command.

The above items list recommended procedures. Here's some recommended tools: `Maple` (expensive), `Xfig` (free), `Mathematica` (expensive), `Gnuplot` (free).

4.5.2 Some other graphics tools

I'll mention here some other tools for producing graphics to use in LaTeX. I've used some of them, but not all, and in case this is entended to just be a list.

- XY-pic. This is a TeX macro package that excels at drawing mathematical diagrams. "Diagrams" are pictures that consist mostly of nodes and things connecting the nodes. The nodes can be simple circles, formulas, points, etc. The things connecting the nodes can be lines, arrows, curves, etc.

- PS-tricks. This a TEX macro package that gives access to PostScript commands within a TEX environment. The syntax is geared to creating the sort of drawings that are typically useful in mathematics.
- potrace. This is a program that can transform a black and white bitmap image (like JPEG, GIF, PNG, etc) into PostScript or PDF, and even X-fig formats.
- pstoedit. This is a program that tranlates PostScript and PDF into other editable formats.

4.5.3 Graphic size

Of course it's a matter of opinion as to what size a graphic should be, and this size depends on where it's going to be placed, as well as how complicated the picture is. Here's some rule's of thumb.

To put a graphic in the margin the width should be 5cm. To put a graphic in a displayed figure, the size should be larger, but it's not clear by how much. As a rule, when people first make graphics they tend to make them too big. A width of 7cm looks fine for most displayed graphics, and probably it should be at most 10cm.

4.5.4 Graphics commands

At present there are small number of custom graphics commands. The command

```
\margingraph{filename}{caption}{label}
```

creates a graphic figure in the margin. It gets the graphic from the file `filename`, it gives it a caption `caption` and it makes a LATEX label `label`. This command at present assumes that the size of the graphic will fit into the margin; although I see no reason not to have it automatically size the graphic to width of the margin.

If one wants to put a graphic in the margin without using the preceding command, one can type

```
\marginpar{\centering\includegraphics{filename}}
```

If one wants a caption in the margin graphic, one can use the command `\figcaption` within the `\marginpar`.

The `free_calc.sty` package loads the `graphicx` package, so all of the commands described in that package can be accessed. However, the internal command `\Ginclude@graphics` has been redefined so that when a `\Show{files}` command has been issued, the name of the graphics file will be printed next to the graphic. This change should not affect the user at all (unless I've made a mistake in the implementation).

4.5.5 Future graphics procedures

At present, all of the graphics have been imported in PDF. This looks quite nice, but there are a few features which make it less than ideal. (1) The fonts come from the program which generate the graphic, rather than LaTeX. (2) Because of this one cannot have formulas. (3) If one resizes the graphic then the lines, fonts, etc are also resized, resulting in lines and fonts that are too big/small.

All of these problems can be avoided, and in the future probably will be. Firstly, if one uses PostScript then the fonts from the graphic generating program can be replaced with LaTeX fonts. The package `psfrag.sty` does this, although it can be done in other ways as well. This solution combines the full power of the graphic generating programs with the nice features of LaTeX. However, re-sizing graphics still changes the thickness of lines, so the original graphic needs to be close to the size that will be finally used. Also, this solution introduces an extra step into the graphic process: first one creates the graphic, then one exports, then one does something to replace the original fonts with LaTeX fonts.

Another solution involves using the package `eepic`. This package greatly enhances the LaTeX picture drawing tools. The program `Xfig` can export directly to `eepic` commands which are contained within a LaTeX environment. This means that the fonts are automatically correct, and that one can edit the file as a LaTeX file in order to position formulas, use macros etc, and that resizing will not affect the lines, fonts etc, as these are generated by LaTeX. The program `Gnuplot` can also export its plots as `eepic` commands. Finally, `Maple` can export plots as a list of points, and the package `coordsys` has a facility for importing these points into a `eepic` environment. The `eepic` solution has two drawbacks: (1) `eepic` cannot reproduce all of the features of `Xfig` so some drawings cannot be fully exported to `eepic`, (2) if one uses `Maple` to export points, which are then combined with `coordsys` and `epic`, and if your plot has more than one function, a separate graphics file is created for each function (I suspect that this feature can be changed in a future version of the tt coordsys package).

I have not yet implemented (or even experimented with) these solutions, but I think that in the future this text will probably adopt one or more of the procedures just described. As mentioned elsewhere, the final format of this text should be in PDF. Thus, all of the procedures just described would require processing the LaTeX source as follows:

$$\text{LaTeX source} \longrightarrow \text{dvi} \longrightarrow \text{PostScript} \longrightarrow \text{PDF}$$

This creates an extra couple of steps as compared to using pdf-LaTeX, however the `psfrag` and `eepic` packages are not compatible with pdf-LaTeX and are not likely to become compatible in the near future. (The extra processing steps required by these packages fits in well with the multi-step procedure involving dvi and PostScript. In contrast, when making a PDF document directly, each page needs to be completely finished and processed at the time the PDF is created.)

4.6 Some re-defined LaTeX commands

I'm sorry, but I've redefined some commands from standard LaTeX and other packages. In general this is a bad idea, but I will keep my changes unless someone can convince me that these particular cases are bad. As far as I can tell, this shouldn't cause problems, but you never know, so here's a list of them.

- `\rule`. In LaTeX this creates a black rectangle. However, I wanted to use "rule" as the name of an environment for giving a **Rule**, like the rule for a derivative. The original LaTeX command has been saved with the new name `\inkrule`. Thus `\inkrule{.5in}{1in}` will make a black rectangle .5in by 1in in size.
- `\begin{proof}` from the `amsthm` package has been re-created. The new `proof` environment should work the same way as the old one.
- `\includegraphics` from the `graphicx` package has been re-defined so that if `\Show{files}` is given then the name of each graphic file will be displayed next to the graphic. In other respects the new version of `\includegraphics` should work the same as the old one.
- `\include` has been re-defined so that it does not issue `\clearpage` commands and so that LaTeX keeps track of which file is currently being processed. In other respects the new version of `\include` should work the same as the old one.
- `\begin{comment}` from the `verbatim` package has been re-defined so that it creates an environment which prints **Comment** which presumably contains comments from the author to the reader. The original command has been saved with the new name `HiddenComment`. Thus

  ```
  \begin{HiddenComment}
  Put some stuff here.

  And more stuff here.
  \end{HiddenComment}
  ```

  will hide all of the material enclosed in the environment.
- `\l` and `\r` are now synonyms for `\left` and `\right` respectively.
- `\i` and `\j` now produce $\hat{\imath}$ and $\hat{\jmath}$.

5 How a hacker uses the text

This section contains specific goals which involve LaTeX coding, or software coding in the usual sense. Of course, you may see other things that you would rather hack, in which case go for it. But, if you're interested in things that I think need fixing, here they are.

- I suspect that a knowledgeable LaTeX programmer would see that much of the LaTeX code I've created in `free_calc.sty` is poorly done, so suggestions and fixes in this regard would be welcome.
- The hyperlinks have not been set up at all. This should be a relatively straight forward project for someone who's interested.
- Various code issuues with the homework.
  - General layout issues for the homework, described in the next major item. Also, need to change where directions for a set of exercises appears.
  - Each exercise should be given in three parts: the problem, a hint, and a solution. Then various options would be set for which parts should display.
  - In addition, it would be nice to have key-values for homework. This would not necessarily be the same key-values that psuedo-paragraphs use. Also, we probably wouldn't want to enter key-values for every problem, but rather sets of problems. So, maybe there would still be an outer psuedo-paragraph like so:

    ```
    \begin{problems}{uses = product_rule}{
    \item $x\sin(x)$
    \item $x^2e^x$
    Etc
    \end{problems}
    ```

    Using this approach would make it easy to implement `\Show{ keys,files}` as well.
- Layout. The notes at present do not look as good as most books. The layout needs to be tuned. There should probably be more than one layout option (i.e. a generic option, an e-book option, etc, stored in the `layouts` directory). Here are a few particulars.
  - We need more distinct colors on most pages. Note: we don't need too many colors, and we don't need them just for entertainment. However, a typical calculus book will have colors which highlight examples, theorems, rules etc. Also, a few colors on every page is more pleasing to the eye than pure black and white.
  - The exercise pages should have a different layout. They should have smaller margins, smaller fonts, and be in twocolumn mode.
  - The numbering of the examples, definitions etc. should be tweaked. In particular, all of these numbers should be reset at the beginning of each section, but this should not use the default LaTeX scheme of numbering as "Example 2.3.11" for chapter 2, section 3, example 11. Rather, it should just be "Example 11", but this should get reset at the beginning of the section.

- Improve the syntax for the Boolean constructions in the `\Show` command. In particular, the requirement of commas everywhere is a bit unnatural and likely to lead to user errors.

- Print out a warning message when someone enters the same "label" key twice. Note that this should be a gentle warning, since there are times where we want the same label to appear twice, at least in editing mode, if not in a final print-out.

- Implement cross-checking. So, if an environment requires limits, it should check if limits have been defined. If an example requires Taylor's Theorem, it should check if Taylor's Theorem has been stated.

- Implement importance levels. Thus, each example could be rated "importance=1", "importance=2", etc. That way, the user could decide to show only the really important stuff.

  Using levels would require discipline. We don't want twenty levels, and everyone will disagree over what's the most important thing. Still, might as well put the option in there.

- Modify the graphics input commands to make them easier and more consistent for the average user.

  In addition, modify the command which shows the name of the graphic file, so that it shows the full path-name, i.e. it shows the name of the directory that the file is contained in.

- Using the unique name-labels, implement commands to show/hide "from...to". Thus, one could say

```
\Show{from=example_limit_of_sin_over_x,
to=example_of_e_to_the_x_over_x}
```

  and this would show those pseudo-paragraphs between the two that are specified by the unique labels.

- Tweak the `\Show` command so that formatting comes out better. Here are some problems with the currrent set-up. Suppose you want to show only theorems. Then the final product will have many empty pages which were generated by `\chapter` and `\section` commands which didn't have any theorems in them.

  The obvious way to fix this is with flags; the psuedo-paragraphs could set a flag which says "this section is not empty", the section and chapter commands would check to see if the section is not empty before displaying the heading. Since the section and chapter commands come before the psuedo-paragraphs, the flag will have to be stored in a `.aux` file, which will record the result from the previous run; the default should be to show.

Thus, if you are showing only theorems, and if chapter 1 has no theorems, on the first run the chapter and section commands would display, this would be followed by empty pages. But on the second run, LaTeX would check the `.aux` file, would see that these sections were empty, and the section and chapter command would not produce anything: it would just gobble it's contents.

A downside of this is that one might have to run LaTeX three times (twice to get rid of the blank pages, and then one more time to make sure the references have stabilized).

- Investigate running this text on `sourceforge`, or as a CVS, or even as a Wiki.

  I've been hesitant to set it up as a Wiki so far, for the following reasons. Firstly, Wikis are presently set up to edit and display documents in html, whereas this project is in LaTeX. Without the use of html, it seems to me that there's no advantage of using a Wiki as compared to some other system with fairly free access. For example, if we set up a CVS and make it really easy for people to have accounts on it, this might be a good thing. Secondly, in the early stages I think this project needs more structure than is usually present in a Wiki. In other words, when the goals are being defined and the framework for adding material is changing, it seems best to me to have an editor/coordinator. Finally, this project involves coding, and code does not lend itself to anonymous/random contributions. There are two levels of code here: the internal code that makes things work, and the user-level code that is used in writing up the material. Changes in the internal code need to be debugged and evaluated before being distributed. As far as the user code goes, I think this could mostly be open to anyone making changes at any time, except for the issue of debugging. I think any change should pass some sort of a test designed to find errors.

- Investigate other markup languanges or software packages for the whole text. Here's a description of one possibility that I'm aware of.

  The language ConTeXt looks like a really advanced approach to automatic typsetting. This package is built on top of TeX, using PDF; one of the main contributors was Hans The Than, the person who invented pdf-TeX. It can be thought of as a parallel to LaTeX: it consists of a large number of macros which describe abstract markup, as opposed to TeX's detail oriented markup.

  ConTeXt is designed to create dynamic documents and address certain deficiencies in LaTeX. The word "dynamic" is used technically, as in contrast to "static". Most documents that you write by hand are static, they will not change except when you change them, and this is probably not often compared to the number of times they will (hopefully) be used. In contrast dynamic documents might change for every user, or might change every day, or might change while you're using them.

These calculus notes represent an attempt to make LaTeX create dynamic documents in a limited way; it has required adding on a lot of packages and code whereas ConTeXt was created from the ground-up to do this sort of thing. ConTeXt can create PDF documents which have interaction; i.e. buttons, forms, conditional execution etc. One drawback of this capabality however, is that the resulting PDF document can not be run on many PDF viewers; it should probably be run on Adobe Acrobat Reader, though in the future other readers may become powerful enough to implement all of these features as well.

Another drawback of ConTeXt is that fewer people know about it, and that the documentation isn't as complete as it is for LaTeX. Here's some links to find out more about ConTeXt.

The home-site of the inventors of ConTeXt:

`http://www.pragma-ade.com/`

The show-case manual (you probably need Adobe Acrobat Reader for this, as opposed to X-pdf, TeXShop, Apple Preview, GSview, Foxit, etc, you also need to download additional filess if you want to follow the links)

`http://www.pragma-ade.com/show-man.pdf`

Here's an useful website which discusses how to migrate LaTeX to ConTeXt:

`http://www.berenddeboer.net/tex/`

- I think it should be fairly easy to put together a simple graphical interface to many of the options in this text.

  All that would be needed is to take some graphical scripting language, ask some questions, write the responses to a configuration file, and then to run pdf-LaTeX in the background. All of this might be possible directly through ConTeXt, however I think it might be best at present to not commit to moving everything to ConTeXt.

  Ultimately, I imagine having a full package on a CD that would work on any computer. The student version of the CD would probably have the broad outline set-up by the teacher. Then the student could go home, start with the "standard" version of the class text, and show only the examples, show extra examples, show extra graphs, etc. An important possibility here is to get hints on the exercises.