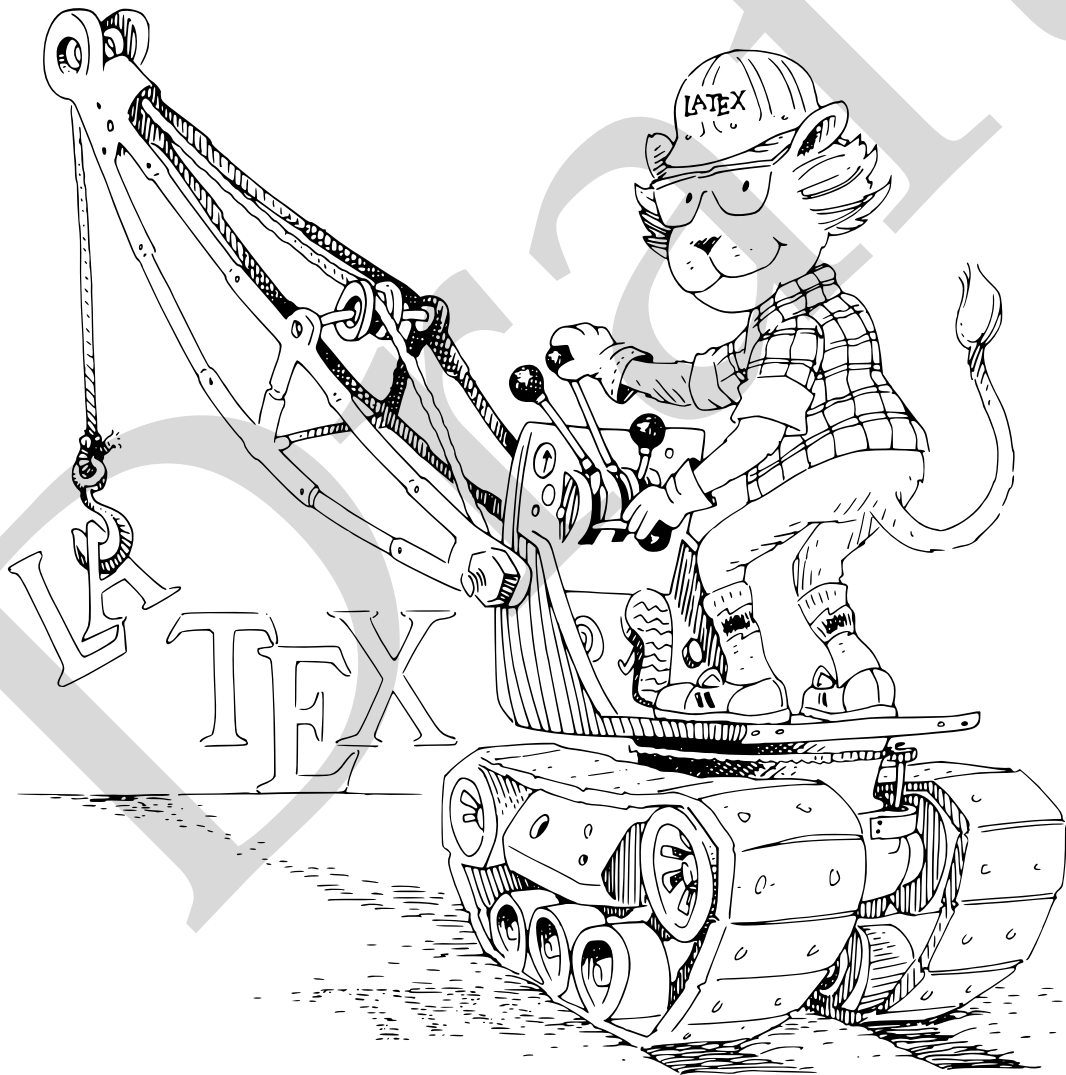


# Sample L<sup>A</sup>T<sub>E</sub>X document

A sample file to showcase L<sup>A</sup>T<sub>E</sub>X

Pieter van den Hombergh  
879417



drawing by Duane Bibby

Draft

# Contents

<b>1 Introduction</b>	<b>1</b>
1.1 Some hints to start with . . . . .	2
<b>2 Motivation</b>	<b>3</b>
2.1 Multiperson literal work . . . . .	3
<b>3 Mathematics to show off</b>	<b>4</b>
3.1 Sums and Integrals . . . . .	4
3.2 Matrices in $\LaTeX$ . . . . .	5
<b>4 Graphics as easy as pie</b>	<b>6</b>
4.1 A png example . . . . .	7
4.2 PDF from an UML package . . . . .	7
<b>5 Listings and code documentation</b>	<b>8</b>
5.1 Source code . . . . .	8
5.2 Makefiles . . . . .	9
<b>6 So <i>you</i> know how to show off, but how do <i>I</i> start?</b>	<b>10</b>
6.1 My own definitions . . . . .	10
<b>7 Drawing in <math>\LaTeX</math></b>	<b>12</b>

# List of Figures

4.1	A Pie chart . . . . .	6
4.2	A class diagram made with Visual Paradigm . . . . .	7
4.3	A class diagram by Umbrello . . . . .	7
7.1	A pgf/Tikz drawing . . . . .	12

Draft

*Introducing oneself properly is always hard.*

Anonymous

# 1

## Introduction

One of the standards for documentation in open source and hence in Linux land is  $\LaTeX$ , a text processing package.  $\LaTeX$  is available for free and available with all Linux distributions and installed in the Lab.

$\TeX$  is the machinery of  $\LaTeX$  and was defined in the  $\TeX$  book [Knuth] and implemented by Prof. Donald Knuth.  $\LaTeX$  is a (nowadays HUGE) set of macros built on top of that.  $\LaTeX$  in its initial form is described by Leslie Lamport in [Lamport]. If you like your book thick, try the  $\LaTeX$  companion [Mittelbach/Goossens].

The web is also a very good source of  $\LaTeX$  documentation. A good starting point [authors] at <http://en.wikibooks.org/wiki/LaTeX>, useful for beginners and pros alike.

This is a simple multi part document. It's purpose is to show how easy it is to create a multi part document, one that, for instance, can be worked on simultaneously by several authors. Note that most of the settings for this document are set in the file `mydefs.tex`. Look in that file too.

This document consists of the following files

1. `main.tex`
2. `motivation.tex`
3. `graphics.tex`
4. `codelisting.tex`
5. `poser.tex`
6. `hello.c`
7. Makefile and
8. `mydefs.tex`
9. `servlet3.png`, the pie chart
10. `Diagram1.pdf`, the UML diagram

You are kindly advised to keep your lab logs in simple text files. These can be turned into latex files easily, which can be used to produce a nice looking report.

## 1.1 Some hints to start with

Sometimes things do not work out the way you think. L<sup>A</sup>T<sub>E</sub>X interpretes some character codes in it's own way. Things like dollar signs or even underscore are special. L<sup>A</sup>T<sub>E</sub>X source are littered with accolades or *curly braces* if that's the way you call them. They are special too. So here is some advice:

### funny file names

Do not use *funny file names*. That is: stick to ASCII filenames without spaces or even underscores. These will bring only you into trouble. If you want to keep things portable, don't use camel case (like in JavaClassNames) either, because some OS-es do not distinguish between upper and lower case. You may of course brake this rule if the files are program things like Java source files.

In software projects, versioning is important. L<sup>A</sup>T<sub>E</sub>X and SubVersion work nicely together here. By using the L<sup>A</sup>T<sub>E</sub>X package `svn-multi` you can have `svn+ latex` insert meta information about your files in the produced output, for instance in the page footers, as in this document.

If you add the codes below at the top of each `.tex` file, these codes will be expanded/updated by `svn on checkin`.

```

\svnidlong {$HeadURL$}%           1
{$LastChangedDate$}%           2
{$LastChangedRevision$}%       3
{$LastChangedBy$}             4
\svnid {$Id$}                  5

```

You must also tell subversion to expand these keywords for those files with for each file the command:

```

25 svn propset svn:keywords "Id _Author _File _Date _LastChangeDate
   _ _Revision _HeadURL _Header" filename

```

To keep these version codes up to date, first check if your L<sup>A</sup>T<sub>E</sub>X files compile, then check them in and do your final L<sup>A</sup>T<sub>E</sub>X run.

To make the version codes of the files appear on the bottom line, have a look at the `mydefs.tex` file where the fancy headers and footers are defined. In group work you may also find it comfortable to create a file named `myauthors.tex` with contents like below, which is then input into the main or `mydefs` at the proper place and can translate student numbers from `svn` and you peerweb account into a humanly readable authors name.

```

\svnRegisterAuthor {879417}{Pieter van den Hombergh} 1

```

What moves, motivates.

Ancient wisdom of technical teachers

# 2

## Motivation

To write a simple document, like a letter, a word processing package is just fine. To automate the creation of a document this is a bit harder. Using a word processor to automatically create documents out of unrelated sources is difficult at best, if doable at all.

However, as long as the only thing that those unrelated sources should produce are simple ASCII documents, things get much more doable. It compares like HTML to Microsoft Word documents. The power you have in defining the layout of a document in HTML (combined with css) with “simple” ASCII document is almost as powerful as what you can do with Word. But now try writing the same thing with a (self written) program. You know it is easy in HTML (certainly if you ever did some programming in e.g. PHP), but producing a Word document with a program is hard work<sup>1</sup>.

If the document should include complex things like mathematical formulas that are layed out properly, it becomes difficult in HTML too.

Enter  $\LaTeX$ .

### 2.1 Multiperson literal work

One of the much overlooked advantages of  $\LaTeX$ (and of any multi-file source code application like java projects) is that the fact that you can split up the document into multiple but still coherent parts. This fact allows you to work on one final document with multiple persons as in team members.

This make  $\LaTeX$  almost ideal for project work in which several authors have to contribute to the final product and where source files are shared by means of a repository. Even more so in cases where you want to include (part of) program source code into the document to explain or show certain implementation aspects. Such source code is not copied and pasted but in stead read on the fly from the original file(s) during text processing. This allows you to always have the most up to date version.

As this sample is a multipart document, it can be used as a reference or a start for your own document.

---

<sup>1</sup>It becomes easier in modern version of word processing packages, they tend to use xml as their internal document format

## Mathematics to show off

Here, you see how a mathematical equation can be generated in line, for instance  $f(x) = \frac{1}{1+25x^2}$ . The  $\$$ -symbols enclose the formula. As a so-called displayed formula, it would look like

$$f(x) = \frac{1}{1 + 25x^2}.$$

It is customary that mathematical functions are *not* set in math-italics, so  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  has the basic ones pre-defined; you should use the commands  $\backslash\cos$ ,  $\backslash\exp$ , etc. to get  $f_1(x) = \cos x$ ,  $f_2(x) = -e^x \sin^2 x$ , etc.

Here, I use some of my commands defined above: I like  $\varepsilon = \varepsilon$  better than the default  $\epsilon$ . A partial derivative (with 2 arguments) would be obtained as follows. If  $f(x, y) = x^2y^3$ , then

$$\frac{\partial f}{\partial x} = 2xy^3, \quad \frac{\partial f}{\partial y} = 3x^2y^2.$$

### 3.1 Sums and Integrals

When you say “capital sigma,” you probably did not really mean  $\Sigma$ , but rather a summation symbol. You would get that as in

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1-r} \quad \text{for all } |r| < 1.$$

Finally, we have

$$\int_0^1 \sin(2\pi x) dx = 0$$

and

$$\iint f(x)g(y) dx dy = \int f(x) dx \int g(y) dy.$$

Here,  $\backslash,$  gives a small space, while  $\backslash!$  forces things closer together; you have to work on the proper spacing for integrals, as  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  does not understand, what is going on.



## 3.2 Matrices in L<sup>A</sup>T<sub>E</sub>X

A matrix  $A \in \mathbb{R}^{m \times n}$  could be defined by

$$A = \begin{pmatrix} 11 & 12 & 13 & \cdots & 1n \\ 21 & 22 & 23 & \cdots & 2n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m1 & m2 & m3 & \cdots & mn \end{pmatrix}$$

Here, the word `dots` in the commands stands for an ellipsis (i.e., three dots) placed horizontally in the centre (`\cdots`), vertically (`\vdots`), or diagonally (`\ddots`); what is not mentioned is `\ldots` for horizontal dots at the baseline. Use the baseline or central version as appropriate, for instance

$$a_1, a_2, \dots, a_n \quad \text{and not} \quad a_1, a_2, \cdots, a_n,$$

$$a_1 + a_2 + \cdots + a_n \quad \text{and not} \quad a_1 + a_2 + \dots + a_n,$$

Some more comments on the matrix are needed, I suppose: The `\left(` and `\right)` create the variable-sized parentheses around the actual array of terms. You can also use `\left[` and `\right]`, or `\left\{` and `\right\}` in other situations. The actual array arrangement is organised by the `array` environment; you need the arguments `ccccc` to indicate that there are five columns and you want the entries centered (“c”), other options are left (“l”) and right (“r”). Notice how `&` separate columns and `\\` the rows.

Here is another matrix example. A matrix multiply used with 3D graphics:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## Graphics as easy as pie

L<sup>A</sup>T<sub>E</sub>X, combined with pdf in `pdflatex`, supports the following graphic file types: pdf, png, jpeg or jpg and gif in that order of preference. Using the vector format pdf gives the added benefit that the graphic file can be scaled up and down without loss of quality. If you want to include bitmaps, try to get them in png format which open and patent free. It has the advantage over jpeg or jpg that it is loss-less, so you do not see any artifact if you blow them up in your inclusion. Converting back from jpg to png is useless, because the damage is already done in the jpeg format. JPEG is excellent for photographs. For all the bitmap formats: try to get them at the intended size with a resolution of 300dpi for printouts. 75 dpi is acceptable for screen reading.

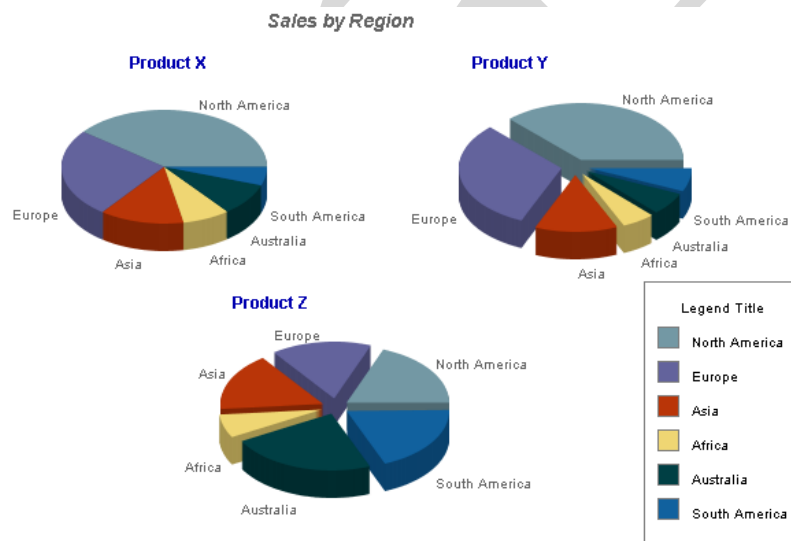


Figure 4.1: Stolen from the net. Google for a pie chart...

Many graphics packages can produce pdf files Embedded postscript (file extension .eps) are also a good candidate, after converting them to pdf with the `epstopdf` tool. By the way: the native format of Adobe Illustrator (ai) is similar enough to eps, so that it can be processed with `ps2pdf`. Programs like *Rational Rose* are able to produce eps files to. And sometimes open-office can lend a helping hand, for by cutting and pasting windows graphics into a single page oodraw drawing, you can produce an very usable pdf file.

Bitmap file types like png and jpeg take up a lot of space in your final pdf document. Bitmap files take even more space if encoded into a pdf file. If at all

possible, stick to a vector format like eps or pdf (if necessary derived from eps files).

## 4.1 A png example

If latex cannot fit the diagram on this page (page 7), then you may find the diagram as figure 4.1 on the facing page. And as you can see, you can easily reference pages and figures.

## 4.2 PDF from an UML package

If the documentation you write is a design document of some software package, you may want to include design diagrams. No software engineering without a UML diagrams... You can see one generated with "dia", a vector drawing program that understands the UML in figure 4.2 on the current page. This diagram is 'wrapped' in a `wrapfigure` environment, so the text may flow around it

The diagram is not very sophisticated but shows an example of a vector format file included via an `eps`→`pdf` conversion by `epstopdf`.

Open source programs like `umlet` and `argouml` are also able to produce vector format graphics files. And sometimes it is helpful to add a box that is a bit bigger than the picture you want to include. This ensures that the so called bounding box does not cut of any lines you want in your picture. Sometimes it is necessary to give these tools a helping hand with `inkscape`, that is do a bit of tinkering to get all details right.

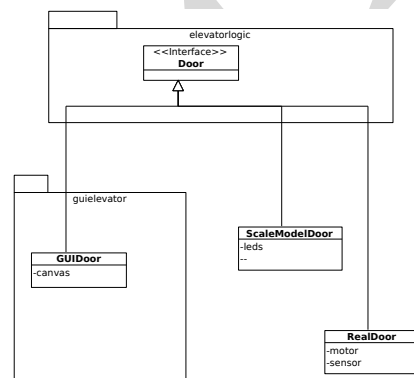


Figure 4.2: A class diagram made with Visual Paradigm

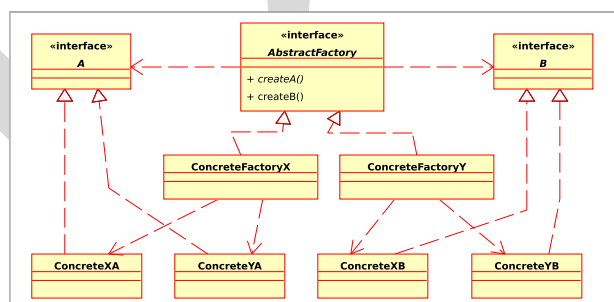


Figure 4.3: A class diagram by Umbrello, saved as `svg` and processed with `inkscape`.

# 5

Use the source Luke.

The most accurate documentation is in the source.

Addendum: Document your source well! And choose proper names.

Pieter van den Hombergh

## Listings and code documentation

For these functions to work you need to use the package listings. See my-defs.tex for the inclusion.

- 5 Note that the line numbers in the right hand border are the line numbers in the included sources.

### doxygen

- A good advice is to start using *doxygen* for your code documentation. It can produce nicely formatted HTML and latex documents and can be tuned in various ways with the help of doxywizard. The way of working is a lot like using javadoc, but also works for C and C++ files. See e.g. the documentation on the zthreads package at <http://zthread.sourceforge.net> or Qt at <http://doc.trolltech.com/3.3/index.html>
- 10

### 5.1 Source code

The most simple case: include the whole thing with a command like

15 `\lstinputlisting[language=java]{code/Hi.java}`

```
1  /**
2   * The obiquiteous Hello World program, a Java variant.
3   * @author Pieter van den Hombergh (879417)
4   */
5  public class Hi{
6      @Override
7      public String toString(){
8          return "Hello_world";
9      }
10     public static void main(String[] args){
11         System.out.println(new Hi());
12     }
13 }
```

Sometimes it is useful to include just a part of a file, for instance when you want to explain things. Like what line 11 is all about.

```
\lstinputlisting[language=java, firstline=11, lastline=11]{Hi.java}
```

Listing 5.1: use toString to let an object represent itself.

```
35 System.out.println(new Hi()); 11
```

## 5.2 Makefiles

You can also include make files. Note that makefiles have a peculiar syntax. *Spaces and tabs in Makefiles* are meaningful. That's why I made them show up in the next listing with the command

### Spaces and tabs in Makefiles

5

```
The make file for this entire document looks like this:
\lstinputlisting [language=make, showtabs=true ,
                 showspaces=true , basicstyle={\ttfamily\scriptsize} ,
```

10

Spaces show up as `␣`, tab characters as an extended version of the same thing (`␣`). As can be expected, spaces and tabs have no special meaning in makefile comments, the lines starting with a hash (#) sign. If you want to know more on makefiles try google or `info:make` in `konqueror` on a decently installed Linux box.

15

The make file for this entire document looks like this:

```
#
#_Makefile_for_the_latex_documentation_of_this_project
#@author_Pieter_van_den_Hombergh_number_879417
#_Id:_Makefile_320_2007-05-17_19:10:36Z_hom_
#_define_some_flags
LATEX=pdflatex
BIBTEX=bibtex
LATEXFLAGS=--interaction=batchmode
#_How_to_produce_the_pdf_from_.tex_files:
#_run_pdfelatex_to
#_get_the_references_right.
#_to_process_a_Latex_doc_into_a_PDF_document
%.pdf:_%_tex
_____$ (LATEX) _$ (LATEXFLAGS) _$<
_____$ (LATEX) _$ (LATEXFLAGS) _$<
_____$ (BIBTEX) _$*
_____$ (LATEX) _$ (LATEXFLAGS) _$<

#_first_some_definitions
TARGET=main.pdf
BARCODE=12345
PARTS=main.tex_mydefs.tex_intro.tex\
____motivation.tex\
____graphics.tex_mathematics.tex\
____codelisting.tex_Makefile_code/hello.c_figures/barcode.pdf\
____poser.tex_frontpage.tex_simple.tex_programmedgraphics.tex\
____figures/servlet3.png_figures/Diagram1.pdf_figures/fon000_00c.pdf

#_things_that_may_be_thrown_away
DISPOSABLES=_*.aux_*.log_*.o_hello

#_define_the_default_target
all: _main.pdf

main.pdf: _main.tex _$ (PARTS)

#_define_a_phony_target
.PHONY: _clean _distclean

clean:
____rm _f _$ (DISPOSABLES)

distclean: _clean
____rm _f _$ (TARGET) _*.toc_*.out_*.url_*.lof
```

60

Never start reading a difficult book at the wrong side.  
It makes you feel stupid.

Private experience

# 6

## So you know how to show off, but how do / start?

This document is indeed a bit of a showcase, but there is more to it. In essence, most documents are mainly texts. And those plain texts take mainly typing and not much more. The minimal, hello world style  $\LaTeX$  file is not much longer<sup>1</sup> than the C classic.

```
1 \svnid{$Id: simple.tex 31 2013-09-08 10:54:29Z 879417 $}
2 \renewcommand\TheFile{simple.tex}
3 \documentclass{report}
4 \begin{document}
5 \chapter{Hi}
6 \pagestyle{empty}
7 Hello world.
8 \end{document}
9 \end{lstlisting}
```

And the pictures, well they are made with other packages, and as long as those can produce a supported format, you can use them.  $\TeX$  and  $\LaTeX$  have an own drawing language, but explaining that would blow up this space. There is a lot of documentation available on  $\TeX$  and  $\LaTeX$  and I could recommend some good books on it. But you need not run off to the nearest book shop. Lots of documentation is on the Net, just as the  $\TeX$  program suite itself.

Look for instance at <http://www.ntg.nl> for the Dutch  $\TeX$  user group and <http://www.dante.de> for the German user group. They are both very alive and kicking.

A very good starting point nowadays is <http://en.wikibooks.org/wiki/LaTeX/>

### 6.1 My own definitions

Standard  $\LaTeX$  output looks a bit dull but are nicely formatted nonetheless. If you want to make the most of your own style for your own or your projects documentation, put your style definitions into a separate file. That way you can keep all definitions of style and *macros* in one spot. This works especially nicely in an multi part document, so the other files (and authors) can concentrate on content.

macros

<sup>1</sup>Counting headers too!

You can *define* your own macros in L<sup>A</sup>T<sub>E</sub>X. As an example a macro, `\define`, **define** which I use to let words stand out at the outer margin. I use it at the first use of a word or concept in the text, to aid the reader in finding the definition. Of course this macro could be extended to put the word into an index. Which makes it a nice exercise. The macro is defined as follows:

```
1 \setlength\footskip{60pt}
2 %%
3 % This macro '\define' puts the argument in em
4 % and in boldface in the margin.
5 \newcommand{\define}[1]{% 1 argument
6   \mbox{}{\textit{#1}}% italics or em
7   \marginpar{\raggedright% no adjust
8     \bfseries\hspace{0pt}#1}% bold
```

and is used as `\define{new word}`.

A drawing created with just a few words.  
Or the old adagium 'a thousand words' reversed.

Private experience

# 7

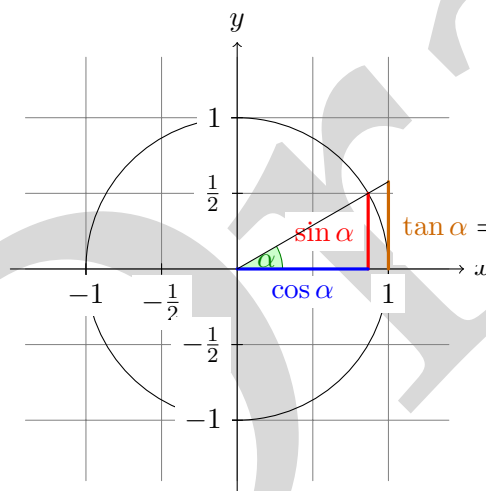
## Drawing in L<sup>A</sup>T<sub>E</sub>X

There are quite a few developers who add usefull packages to the T<sub>E</sub>Xworld. One of them is Till Tantau of the "Technische Universität" in Berlin Germany.  
5 He produced the package called pgf, which stands for portable graphics format.

The pgf package contains the macro tool tikz, that allows you to draw quite nice graphics with just a few commands. Like this small ellipse:  $\circ$

The pgf manual has many nice examples that may prove usefull, certainly if you want to use the package beamer<sup>1</sup> to create professional looking pdf presentations  
10 with L<sup>A</sup>T<sub>E</sub>X.

A simple figure from the pgf manual:



The angle  $\alpha$  is  $30^\circ$  in the example ( $\pi/6$  in radians). The sine of  $\alpha$ , which is the height of the red line, is

$$\sin \alpha = 1/2.$$

By the Theorem of Pythagoras ...

Figure 7.1: A drawing taken from the pgf manual/tutorial by Till Tantau

<sup>1</sup>also by Till Tantau



# Bibliography

**authors, Various:** LaTeX - Wikibooks, open books for an open world. 2011 ⟨URL: <http://en.wikibooks.org/wiki/LaTeX>⟩

**Knuth, Donald E.:** The TeXbook. Addison-Wesley, 1984

5 **Lamport, Leslie:** L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System. Addison-Wesley, 1986

**Mittelbach, Frank/Goossens, Michel:** The L<sup>A</sup>T<sub>E</sub>X Companion Second Edition. Addison-Wesley, 2004

Draft

	svnkeywords.head . . . . .	2
	myauthors.tex . . . . .	2
	code/Hi.java . . . . .	8
5	5.1 use toString to let an object represent itself. . . . .	8
	codelisting.tex . . . . .	9
	Makefile . . . . .	9
	simple.tex . . . . .	10
	mydefs.tex . . . . .	11

Draft

Draft

Draft

Draft

