

Ti*k*Z
L!KΣ

&
&

PGF
BCE

Manual for Version 3.1.5b

MANUAL for Version 3.1.5b

```
\begin{tikzpicture}
  \coordinate (front) at (0,0);
  \coordinate (horizon) at (0,.31\paperheight);
  \coordinate (bottom) at (0,-.6\paperheight);
  \coordinate (sky) at (0,.57\paperheight);
  \coordinate (left) at (-.51\paperwidth,0);
  \coordinate (right) at (.51\paperwidth,0);

  \shade [bottom color=white,
    top color=blue!30!black!50]
    ([yshift=-5mm]horizon -| left)
    rectangle (sky -| right);

  \shade [bottom color=black!70!green!25,
    top color=black!70!green!10]
    (front -| left) -- (horizon -| left)
    decorate [decoration=random steps] {
      -- (horizon -| right)
    }
    -- (front -| right) -- cycle;

  \shade [top color=black!70!green!25,
    bottom color=black!25]
    ([yshift=-5mm-1pt]front -| left)
    rectangle ([yshift=1pt]front -| right);

  \fill [black!25]
    (bottom -| left)
    rectangle ([yshift=-5mm]front -| right);

  \def\nodeshadowed[#1]#2;{
    \node[scale=2,above,#1]{
      \global\setbox\mybox=\hbox{#2}
      \copy\mybox};
    \node[scale=2,above,#1,yscale=-1,
      scope fading=south,opacity=0.4]{\box\mybox};
  }
```

```
\nodeshadowed [at={(-5,8 )},yslant=0.05]
  {\Huge Ti\textcolor{orange}{\emph{k}}Z};
\nodeshadowed [at={( 0,8.3)}]
  {\huge \textcolor{green!50!black!50}{\&}};
\nodeshadowed [at={( 5,8 )},yslant=-0.05]
  {\Huge \textsc{PGF}};
\nodeshadowed [at={( 0,5 )}]
  {Manual for Version \pgftypesetversion};

\foreach \where in {-9cm,9cm} {
  \nodeshadowed [at={(\where,5cm)}] {\tikz
    \draw [green!20!black, rotate=90,
      l-system={rule set={F -> FF-[-F+F]+[+F-F]},
        axiom=F, order=4, step=2pt,
        randomize step percent=50, angle=30,
        randomize angle percent=5}] l-system; }}

\foreach \i in {0.5,0.6,...,2}
  \fill
    [white,opacity=\i/2,
    decoration=Koch snowflake,
    shift=(horizon),shift={(\rand*11,rnd*7)},
    scale=\i,double copy shadow={
      opacity=0.2,shadow xshift=0pt,
      shadow yshift=3*\i pt,fill=white,draw=none}]
    decorate {
      decorate {
        decorate {
          (0,0) - ++(60:1) -- ++(-60:1) -- cycle
        }
      }
    };

\node (left text) ...
\node (right text) ...

\fill [decorate,decoration={footprints,foot of=gnome},
  opacity=.5,brown] (rand*8,-rnd*10)
  to [out=rand*180,in=rand*180] (rand*8,-rnd*10);
\end{tikzpicture}
```

Für meinen Vater, damit er noch viele schöne T_EX-Graphiken erschaffen kann.

Till

Copyright 2007 to 2013 by Till Tantau

Permission is granted to copy, distribute and/or modify *the documentation* under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

Permission is granted to copy, distribute and/or modify *the code of the package* under the terms of the GNU Public License, Version 2 or any later version published by the Free Software Foundation. A copy of the license is included in the section entitled GNU Public License.

Permission is also granted to distribute and/or modify *both the documentation and the code* under the conditions of the LaTeX Project Public License, either version 1.3 of this license or (at your option) any later version. A copy of the license is included in the section entitled L^AT_EX Project Public License.

The TikZ and PGF Packages

Manual for version 3.1.5b

<https://github.com/pgf-tikz/pgf>

Till Tantau*

Institut für Theoretische Informatik
Universität zu Lübeck

July 16, 2020

Contents



1	Introduction	5
1.1	The Layers Below TikZ	5
1.2	Comparison with Other Graphics Packages	6
1.3	Utility Packages	6
1.4	How to Read This Manual	7
1.5	Authors and Acknowledgements	7
1.6	Getting Help	7
I	Tutorials and Guidelines	7
II	Installation and Configuration	8
III	TikZ ist <i>kein</i> Zeichenprogramm	9
IV	Graph Drawing	10
V	Libraries	10
VI	Data Visualization	11
VII	Utilities	12
VIII	Mathematical and Object-Oriented Engines	13
IX	The Basic Layer	14
X	The System Layer	15

*Editor of this documentation. Parts of this documentation have been written by other authors as indicated in these parts or chapters and in Section 1.5.

XI	References and Index	16
	Index	18

1 Introduction

Welcome to the documentation of TikZ and the underlying PGF system. What began as a small L^AT_EX style for creating the graphics in my (Till Tantau's) PhD thesis directly with pdfL^AT_EX has now grown to become a full-blown graphics language with a manual of over a thousand pages. The wealth of options offered by TikZ is often daunting to beginners; but fortunately this documentation comes with a number slowly-paced tutorials that will teach you almost all you should know about TikZ without your having to read the rest.

I wish to start with the questions “What is TikZ?” Basically, it just defines a number of T_EX commands that draw graphics. For example, the code `\tikz \draw (0pt,0pt) --(20pt,6pt);` yields the line  and the code `\tikz \fill[orange] (1ex,1ex) circle (1ex);` yields . In a sense, when you use TikZ you “program” your graphics, just as you “program” your document when you use T_EX. This also explains the name: TikZ is a recursive acronym in the tradition of “GNU’s Not Unix” and means “TikZ ist *kein* Zeichenprogramm”, which translates to “TikZ is not a drawing program”, cautioning the reader as to what to expect. With TikZ you get all the advantages of the “T_EX-approach to typesetting” for your graphics: quick creation of simple graphics, precise positioning, the use of macros, often superior typography. You also inherit all the disadvantages: steep learning curve, no WYSIWYG, small changes require a long recompilation time, and the code does not really “show” how things will look like.

Now that we know what TikZ is, what about “PGF”? As mentioned earlier, TikZ started out as a project to implement T_EX graphics macros that can be used both with pdfL^AT_EX and also with the classical (PostScript-based) L^AT_EX. In other words, I wanted to implement a “portable graphics format” for T_EX – hence the name PGF. These early macros are still around and they form the “basic layer” of the system described in this manual, but most of the interaction an author has these days is with TikZ – which has become a whole language of its own.

1.1 The Layers Below TikZ

It turns out that there are actually *two* layers below TikZ:

System layer: This layer provides a complete abstraction of what is going on “in the driver”. The driver is a program like `dvips` or `dvipdfm` that takes a `.dvi` file as input and generates a `.ps` or a `.pdf` file. (The `pdftex` program also counts as a driver, even though it does not take a `.dvi` file as input. Never mind.) Each driver has its own syntax for the generation of graphics, causing headaches to everyone who wants to create graphics in a portable way. PGF’s system layer “abstracts away” these differences. For example, the system command `\pgfsys@lineto{10pt}{10pt}` extends the current path to the coordinate (10pt,10pt) of the current `{pgfpicture}`. Depending on whether `dvips`, `dvipdfm`, or `pdftex` is used to process the document, the system command will be converted to different `\special` commands. The system layer is as “minimalistic” as possible since each additional command makes it more work to port PGF to a new driver.

As a user, you will not use the system layer directly.

Basic layer: The basic layer provides a set of basic commands that allow you to produce complex graphics in a much easier manner than by using the system layer directly. For example, the system layer provides no commands for creating circles since circles can be composed from the more basic Bézier curves (well, almost). However, as a user you will want to have a simple command to create circles (at least I do) instead of having to write down half a page of Bézier curve support coordinates. Thus, the basic layer provides a command `\pgfpathcircle` that generates the necessary curve coordinates for you.

The basic layer consists of a *core*, which consists of several interdependent packages that can only be loaded *en bloc*, and additional *modules* that extend the core by more special-purpose commands like node management or a plotting interface. For instance, the BEAMER package uses only the core and not, say, the `shapes` modules.

In theory, TikZ itself is just one of several possible “frontends”. which are sets of commands or a special syntax that makes using the basic layer easier. A problem with directly using the basic layer is that code written for this layer is often too “verbose”. For example, to draw a simple triangle, you may need as many as five commands when using the basic layer: One for beginning a path at the first corner of the triangle, one for extending the path to the second corner, one for going to the third, one for closing the path, and one for actually painting the triangle (as opposed to filling it). With the TikZ frontend all this boils down to a single simple METAFONT-like command:

```
\draw (0,0) -- (1,0) -- (1,1) -- cycle;
```

In practice, `TikZ` is the only “serious” frontend for PGF. It gives you access to all features of PGF, but it is intended to be easy to use. The syntax is a mixture of METAFONT and PSTricks and some ideas of myself. There are other frontends besides `TikZ`, but they are intended more as “technology studies” and less as serious alternatives to `TikZ`. In particular, the `pgfpict2e` frontend reimplements the standard L^AT_EX `{picture}` environment and commands like `\line` or `\vector` using the PGF basic layer. This layer is not really “necessary” since the `pict2e.sty` package does at least as good a job at reimplementing the `{picture}` environment. Rather, the idea behind this package is to have a simple demonstration of how a frontend can be implemented.

Since most users will only use `TikZ` and almost no one will use the system layer directly, this manual is mainly about `TikZ` in the first parts; the basic layer and the system layer are explained at the end.

1.2 Comparison with Other Graphics Packages

`TikZ` is not the only graphics package for T_EX. In the following, I try to give a reasonably fair comparison of `TikZ` and other packages.

1. The standard L^AT_EX `{picture}` environment allows you to create simple graphics, but little more. This is certainly not due to a lack of knowledge or imagination on the part of L^AT_EX’s designer(s). Rather, this is the price paid for the `{picture}` environment’s portability: It works together with all backend drivers.
2. The `pstricks` package is certainly powerful enough to create any conceivable kind of graphic, but it is not really portable. Most importantly, it does not work with `pdftex` nor with any other driver that produces anything but PostScript code.

Compared to `TikZ`, `pstricks` has a similar support base. There are many nice extra packages for special purpose situations that have been contributed by users over the last decade. The `TikZ` syntax is more consistent than the `pstricks` syntax as `TikZ` was developed “in a more centralized manner” and also “with the shortcomings on `pstricks` in mind”.

3. The `xypic` package is an older package for creating graphics. However, it is more difficult to use and to learn because the syntax and the documentation are a bit cryptic.
4. The `dratex` package is a small graphic package for creating a graphics. Compared to the other package, including `TikZ`, it is very small, which may or may not be an advantage.
5. The `metapost` program is a powerful alternative to `TikZ`. It used to be an external program, which entailed a bunch of problems, but in LuaT_EX it is now built in. An obstacle with `metapost` is the inclusion of labels. This is *much* easier to achieve using PGF.
6. The `xfig` program is an important alternative to `TikZ` for users who do not wish to “program” their graphics as is necessary with `TikZ` and the other packages above. There is a conversion program that will convert `xfig` graphics to `TikZ`.

1.3 Utility Packages

The PGF package comes along with a number of utility package that are not really about creating graphics and which can be used independently of PGF. However, they are bundled with PGF, partly out of convenience, partly because their functionality is closely intertwined with PGF. These utility packages are:

1. The `pgfkeys` package defines a powerful key management facility. It can be used completely independently of PGF.
2. The `pgffor` package defines a useful `\foreach` statement.
3. The `pgfcalendar` package defines macros for creating calendars. Typically, these calendars will be rendered using PGF’s graphic engine, but you can use `pgfcalendar` also typeset calendars using normal text. The package also defines commands for “working” with dates.
4. The `pgfpages` package is used to assemble several pages into a single page. It provides commands for assembling several “virtual pages” into a single “physical page”. The idea is that whenever T_EX has a page ready for “shipout”, `pgfpages` interrupts this shipout and instead stores the page to be shipped out in a special box. When enough “virtual pages” have been accumulated in this way, they are scaled

down and arranged on a “physical page”, which then *really* shipped out. This mechanism allows you to create “two page on one page” versions of a document directly inside \LaTeX without the use of any external programs. However, `pgfpages` can do quite a lot more than that. You can use it to put logos and watermark on pages, print up to 16 pages on one page, add borders to pages, and more.

1.4 How to Read This Manual

This manual describes both the design of `TikZ` and its usage. The organization is very roughly according to “user-friendliness”. The commands and subpackages that are easiest and most frequently used are described first, more low-level and esoteric features are discussed later.

If you have not yet installed `TikZ`, please read the installation first. Second, it might be a good idea to read the tutorial. Finally, you might wish to skim through the description of `TikZ`. Typically, you will not need to read the sections on the basic layer. You will only need to read the part on the system layer if you intend to write your own frontend or if you wish to port PGF to a new driver.

The “public” commands and environments provided by the system are described throughout the text. In each such description, the described command, environment or option is printed in red. Text shown in green is optional and can be left out.

1.5 Authors and Acknowledgements

The bulk of the PGF system and its documentation was written by Till Tantau. A further member of the main team is Mark Wibrow, who is responsible, for example, for the PGF mathematical engine, many shapes, the decoration engine, and matrices. The third member is Christian Feuersänger who contributed the floating point library, image externalization, extended key processing, and automatic hyperlinks in the manual.

Furthermore, occasional contributions have been made by Christophe Jorssen, Jin-Hwan Cho, Olivier Binda, Matthias Schulz, Renée Ahrens, Stephan Schuster, and Thomas Neumann.

Additionally, numerous people have contributed to the PGF system by writing emails, spotting bugs, or sending libraries and patches. Many thanks to all these people, who are too numerous to name them all!

1.6 Getting Help

When you need help with PGF and `TikZ`, please do the following:

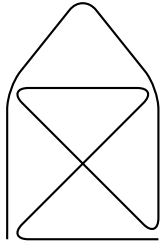
1. Read the manual, at least the part that has to do with your problem.
2. If that does not solve the problem, try having a look at the GitHub development page for PGF and `TikZ` (see the title of this document). Perhaps someone has already reported a similar problem and someone has found a solution.
3. On the website you will find numerous forums for getting help. There, you can write to help forums, file bug reports, join mailing lists, and so on.
4. Before you file a bug report, especially a bug report concerning the installation, make sure that this is really a bug. In particular, have a look at the `.log` file that results when you \TeX your files. This `.log` file should show that all the right files are loaded from the right directories. Nearly all installation problems can be resolved by looking at the `.log` file.
5. *As a last resort* you can try to email me (Till Tantau) or, if the problem concerns the mathematical engine, Mark Wibrow. I do not mind getting emails, I simply get way too many of them. Because of this, I cannot guarantee that your emails will be answered in a timely fashion or even at all. Your chances that your problem will be fixed are somewhat higher if you mail to the PGF mailing list (naturally, I read this list and answer questions when I have the time).

Part I

Tutorials and Guidelines

by Till Tantau

To help you get started with TikZ, instead of a long installation and configuration section, this manual starts with tutorials. They explain all the basic and some of the more advanced features of the system, without going into all the details. This part also contains some guidelines on how you should proceed when creating graphics using TikZ.



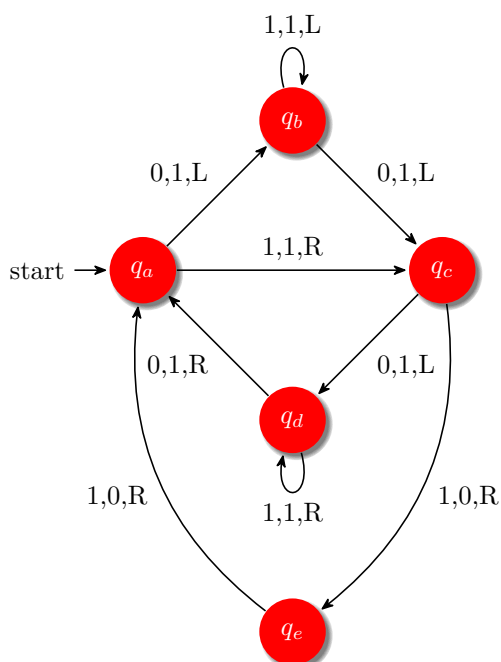
```
\tikz \draw[thick,rounded corners=8pt]
(0,0) -- (0,2) -- (1,3.25) -- (2,2) -- (2,0) -- (0,2) -- (2,2) -- (0,0) -- (2,0);
```

Part II

Installation and Configuration

by Till Tantau

This part explains how the system is installed. Typically, someone has already done so for your system, so this part can be skipped; but if this is not the case and you are the poor fellow who has to do the installation, read the present part.



The current candidate for the busy beaver for five states. It is presumed that this Turing machine writes a maximum number of 1's before halting among all Turing machines with five states and the tape alphabet $\{0, 1\}$. Proving this conjecture is an open research problem.


```

\usetikzlibrary {arrows.meta,automata,positioning,shadows}
\begin{tikzpicture}[->,>={Stealth[round]},shorten >=1pt,auto,node distance=2.8cm,on grid,semithick,
every state/.style={fill=red,draw=none,circular drop shadow,text=white}]

\node[initial,state] (A) {} {$q_a$};
\node[state] (B) [above right=of A] {$q_b$};
\node[state] (D) [below right=of A] {$q_d$};
\node[state] (C) [below right=of B] {$q_c$};
\node[state] (E) [below=of D] {$q_e$};

\path (A) edge node {0,1,L} (B)
edge node {1,1,R} (C)
(B) edge [loop above] node {1,1,L} (B)
edge node {0,1,L} (C)
(C) edge node {0,1,L} (D)
edge [bend left] node {1,0,R} (E)
(D) edge [loop below] node {1,1,R} (D)
edge node {0,1,R} (A)
(E) edge [bend left] node {1,0,R} (A);

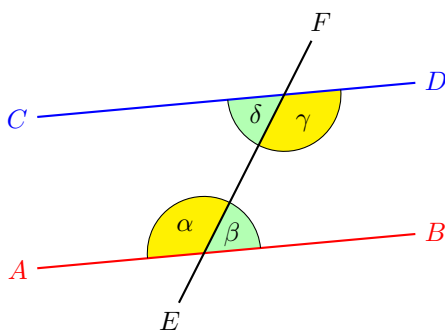
\node [right=1cm,text width=8cm] at (C)
{
The current candidate for the busy beaver for five states. It is
presumed that this Turing machine writes a maximum number of
$1$'s before halting among all Turing machines with five states
and the tape alphabet $\{0, 1\}$. Proving this conjecture is an
open research problem.
};
\end{tikzpicture}

```

Part III

TikZ ist *kein* Zeichenprogramm

by Till Tantau



When we assume that AB and CD are parallel, i. e., $AB \parallel CD$, then $\alpha = \gamma$ and $\beta = \delta$.

```

\usetikzlibrary {angles,calc,quotes}
\begin{tikzpicture}[angle radius=.75cm]

\node (A) at (-2,0) [red,left] {$A$};
\node (B) at ( 3,.5) [red,right] {$B$};
\node (C) at (-2,2) [blue,left] {$C$};
\node (D) at ( 3,2.5) [blue,right] {$D$};
\node (E) at (60:-5mm) [below] {$E$};
\node (F) at (60:3.5cm) [above] {$F$};

\coordinate (X) at (intersection cs:first line={A}--(B), second line={E}--(F));
\coordinate (Y) at (intersection cs:first line={C}--(D), second line={E}--(F));

\path
(A) edge [red, thick] (B)
(C) edge [blue, thick] (D)
(E) edge [thick] (F)
pic ["$\alpha$", draw, fill=yellow] {angle = F--X--A}
pic ["$\beta$", draw, fill=green!30] {angle = B--X--F}
pic ["$\gamma$", draw, fill=yellow] {angle = E--Y--D}
pic ["$\delta$", draw, fill=green!30] {angle = C--Y--E};

\node at ($ (D)! .5!(B) $) [right=1cm,text width=6cm,rounded corners,fill=red!20,inner sep=1ex]
{
  When we assume that $\color{red}AB$ and $\color{blue}CD$ are
  parallel, i.\,e., $\color{red}AB \parallel \color{blue}CD$,
  then $\alpha = \gamma$ and $\beta = \delta$.
};
\end{tikzpicture}

```

Part IV

Graph Drawing

by Till Tantau et al.

Graph drawing algorithms do the tough work of computing a layout of a graph for you. *TikZ* comes with powerful such algorithms, but you can also implement new algorithms in the Lua programming language.

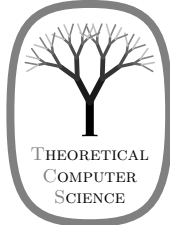
You need to use *LuaTeX* to typeset this part of the manual (and, also, to use algorithmic graph drawing).

Part V

Libraries

by Till Tantau

In this part the library packages are documented. They provide additional predefined graphic objects like new arrow heads or new plot marks, but sometimes also extensions of the basic PGF or *TikZ* system. The libraries are not loaded by default since many users will not need them.



```
\usetikzlibrary {arrows,trees}
\tikzset{
  ld/.style={level distance=#1},lw/.style={line width=#1},
  level 1/.style={ld=4.5mm, trunk, lw=1ex, sibling angle=60},
  level 2/.style={ld=3.5mm, trunk!80!leaf a,lw=.8ex,sibling angle=56},
  level 3/.style={ld=2.75mm, trunk!60!leaf a,lw=.6ex,sibling angle=52},
  level 4/.style={ld=2mm, trunk!40!leaf a,lw=.4ex,sibling angle=48},
  level 5/.style={ld=1mm, trunk!20!leaf a,lw=.3ex,sibling angle=44},
  level 6/.style={ld=1.75mm, leaf a, lw=.2ex,sibling angle=40},
}
\pgfarrowsdeclare{leaf}{leaf}
{ \pgfarrowslefttextend{-2pt} \pgfarrowsrighttextend{1pt}}
{
  \pgfpathmoveto{\pgfpoint{-2pt}{0pt}}
  \pgfpatharc{150}{30}{1.8pt}
  \pgfpatharc{-30}{-150}{1.8pt}
  \pgfusepathqfill
}

\newcommand{\logo}[5]
{
  \colorlet{border}{#1}
  \colorlet{trunk}{#2}
  \colorlet{leaf a}{#3}
  \colorlet{leaf b}{#4}
  \begin{tikzpicture}
    \scriptsize\scshape
    \draw[border,line width=1ex,yshift=.3cm,
      yscale=1.45,xscale=1.05,looseness=1.42]
      (1,0) to [out=90, in=0] (0,1) to [out=180,in=90] (-1,0)
      to [out=-90,in=-180] (0,-1) to [out=0, in=-90] (1,0) -- cycle;

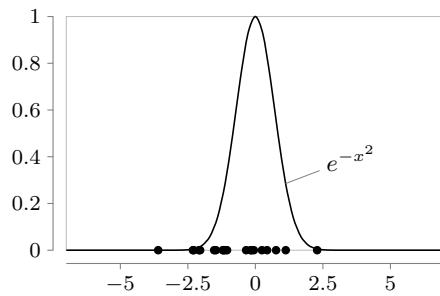
    \coordinate (root) [grow cyclic,rotate=90]
    child {
      child [line cap=round] foreach \a in {0,1} {
        child foreach \b in {0,1} {
          child foreach \c in {0,1} {
            child foreach \d in {0,1} {
              child foreach \leafcolor in {leaf a,leaf b}
                { edge from parent [color=\leafcolor,-#5] }
            } } }
          } edge from parent [shorten >=-1pt,serif cm-,line cap=butt]
        };

        \node [align=center,below] at (0pt,-.5ex)
        { \textcolor{border}{T}heoretical \ \ \textcolor{border}{C}omputer \ \
          \textcolor{border}{S}cience };
      \end{tikzpicture}
    }
  \begin{minipage}{3cm}
    \logo{green!80!black}{green!25!black}{green}{green!80}{leaf}\\
    \logo{green!50!black}{black}{green!80!black}{red!80!green}{leaf}\\
    \logo{red!75!black}{red!25!black}{red!75!black}{orange}{leaf}\\
    \logo{black!50}{black}{black!50}{black!25}{}
  \end{minipage}
}
```

Part VI

Data Visualization

by Till Tantau



• $\sum_{i=1}^{10} x_i$, where $x_i \sim U(-1, 1)$

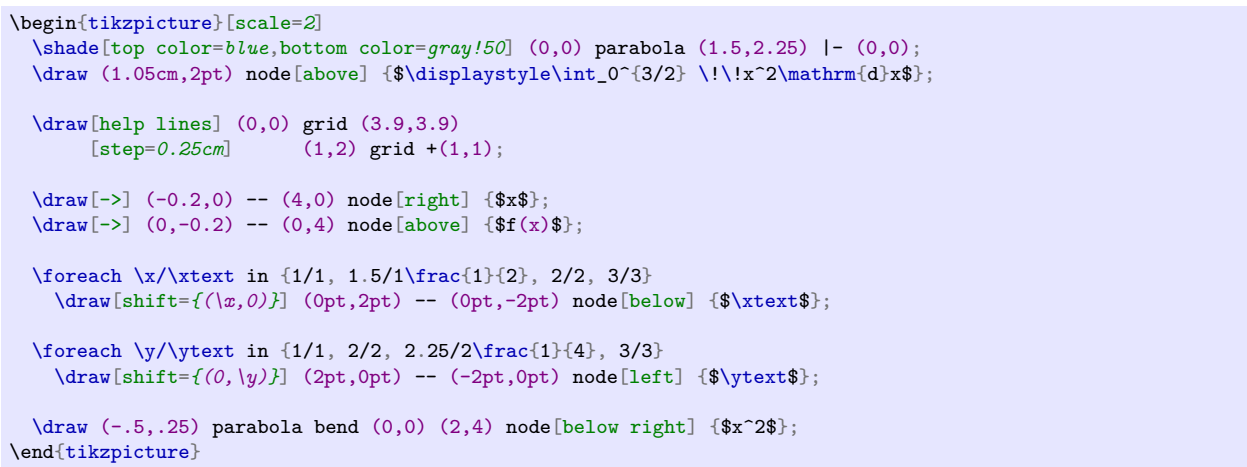
```
\usetikzlibrary {datavisualization.formats.functions}
\tikz \datavisualization [scientific axes=clean]
[
  visualize as smooth line=Gaussian,
  Gaussian={pin in data={text={\mathit{e}^{-x^2}}},when=x is 1}}
]
data [format=function] {
  var x : interval [-7:7] samples 51;
  func y = exp(-\value x*\value x);
}
[
  visualize as scatter,
  legend={south east outside},
  scatter={
    style={mark=*,mark size=1.4pt},
    label in legend={text={
      $\sum_{i=1}^{10} x_i$, where $x_i \sim U(-1,1)$ }}
  }
]
data [format=function] {
  var i : interval [0:1] samples 20;
  func y = 0;
  func x = (rand + rand + rand + rand + rand +
    rand + rand + rand + rand + rand);
};
```

Part VII

Utilities

by Till Tantau

The utility packages are not directly involved in creating graphics, but you may find them useful nonetheless. All of them either directly depend on PGF or they are designed to work well together with PGF even though they can be used in a stand-alone way.





```

\pgfmathsetseed{1}
\foreach \col in {black,red,green,blue}
{
  \begin{tikzpicture}[x=10pt,y=10pt,ultra thick,baseline,line cap=round]
    \coordinate (current point) at (0,0);
    \coordinate (old velocity) at (0,0);
    \coordinate (new velocity) at (rand,rand);

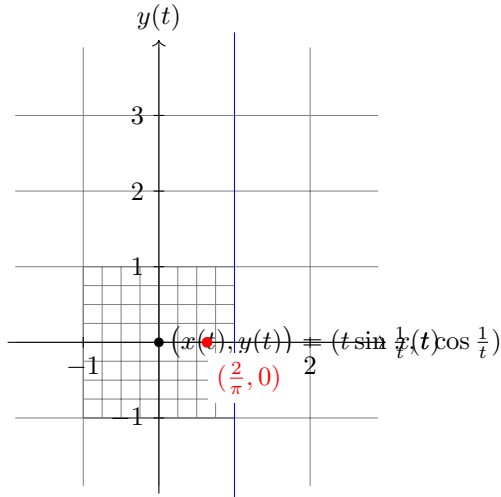
    \foreach \i in {0,1,...,100}
    {
      \draw[\col!\i] (current point)
        .. controls ++([scale=-1]old velocity) and
          ++(new velocity) .. ++(rand,rand)
        coordinate (current point);
      \coordinate (old velocity) at (new velocity);
      \coordinate (new velocity) at (rand,rand);
    }
  \end{tikzpicture}
}

```

Part IX

The Basic Layer

by Till Tantau



```
\begin{tikzpicture}
  \draw[gray,very thin] (-1.9,-1.9) grid (2.9,3.9)
    [step=0.25cm] (-1,-1) grid (1,1);
  \draw[blue] (1,-2.1) -- (1,4.1); % asymptote

  \draw[->] (-2,0) -- (3,0) node[right] {$x(t)$};
  \draw[->] (0,-2) -- (0,4) node[above] {$y(t)$};

  \foreach \pos in {-1,2}
    \draw[shift={(\pos,0)}] (0pt,2pt) -- (0pt,-2pt) node[below] {$\pos$};

  \foreach \pos in {-1,1,2,3}
    \draw[shift={(0,\pos)}] (2pt,0pt) -- (-2pt,0pt) node[left] {$\pos$};

  \fill (0,0) circle (0.064cm);
  \draw[thick,parametric,domain=0.4:1.5,samples=200]
    % The plot is reparameterised such that there are more samples
    % near the center.
    plot[id=asymptotic-example] function{(t*t*t)*sin(1/(t*t*t)),(t*t*t)*cos(1/(t*t*t))}
    node[right] {$\bigl(x(t),y(t)\bigr) = (t\sin \frac{1}{t}, t\cos \frac{1}{t})$};

  \fill[red] (0.63662,0) circle (2pt)
    node [below right,fill=white,yshift=-4pt] {$(\frac{2}{\pi},0)$};
\end{tikzpicture}
```

Part X

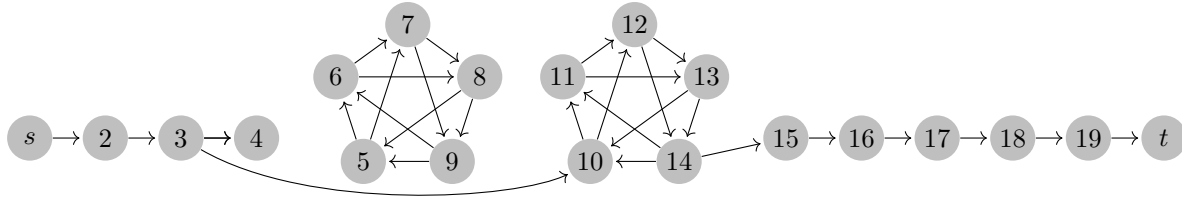
The System Layer

by Till Tantau

This part describes the low-level interface of PGF, called the *system layer*. This interface provides a complete abstraction of the internals of the underlying drivers.

Unless you intend to port PGF to another driver or unless you intend to write your own optimized frontend, you need not read this part.

In the following it is assumed that you are familiar with the basic workings of the `graphics` package and that you know what \TeX -drivers are and how they work.



```

\begin{tikzpicture}
[shorten >=1pt,->,
 vertex/.style={circle,fill=black!25,minimum size=17pt,inner sep=0pt}]

\foreach \name/\x in {s/1, 2/2, 3/3, 4/4, 15/11, 16/12, 17/13, 18/14, 19/15, t/16}
\node[vertex] (G-\name) at (\x,0) {$\name$};

\foreach \name/\angle/\text in {P-1/234/5, P-2/162/6, P-3/90/7, P-4/18/8, P-5/-54/9}
\node[vertex,xshift=6cm,yshift=.5cm] (\name) at (\angle:1cm) {$\text$};

\foreach \name/\angle/\text in {Q-1/234/10, Q-2/162/11, Q-3/90/12, Q-4/18/13, Q-5/-54/14}
\node[vertex,xshift=9cm,yshift=.5cm] (\name) at (\angle:1cm) {$\text$};

\foreach \from/\to in {s/2,2/3,3/4,3/4,15/16,16/17,17/18,18/19,19/t}
\draw (G-\from) -- (G-\to);

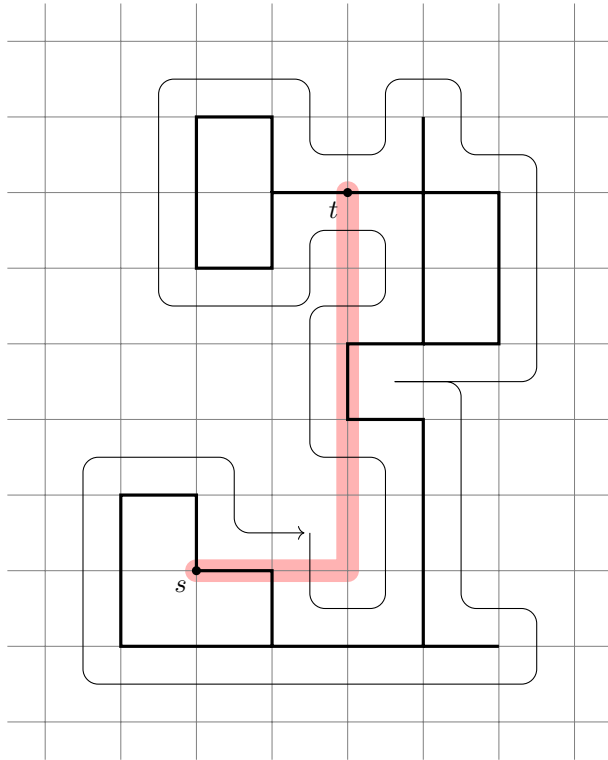
\foreach \from/\to in {1/2,2/3,3/4,4/5,5/1,1/3,2/4,3/5,4/1,5/2}
{ \draw (P-\from) -- (P-\to); \draw (Q-\from) -- (Q-\to); }

\draw (G-3) .. controls +(-30:2cm) and +(-150:1cm) .. (Q-1);
\draw (Q-5) -- (G-15);
\end{tikzpicture}

```


Part XI

References and Index



```
\begin{tikzpicture}
\draw[line width=0.3cm,color=red!30,line cap=round,line join=round] (0,0)--(2,0)--(2,5);
\draw[help lines] (-2.5,-2.5) grid (5.5,7.5);
\draw[very thick] (1,-1)--(-1,-1)--(-1,1)--(0,1)--(0,0)--
(1,0)--(1,-1)--(3,-1)--(3,2)--(2,2)--(2,3)--(3,3)--
(3,5)--(1,5)--(1,4)--(0,4)--(0,6)--(1,6)--(1,5)
(3,3)--(4,3)--(4,5)--(3,5)--(3,6)
(3,-1)--(4,-1);
\draw[below left] (0,0) node(s){$s$};
\draw[below left] (2,5) node(t){$t$};
\fill (0,0) circle (0.06cm) (2,5) circle (0.06cm);
\draw[->,rounded corners=0.2cm,shorten >=2pt]
(1.5,0.5)-- ++(0,-1)-- ++(1,0)-- ++(0,2)-- ++(-1,0)-- ++(0,2)-- ++(1,0)--
++(0,1)-- ++(-1,0)-- ++(0,-1)-- ++(-2,0)-- ++(0,3)-- ++(2,0)-- ++(0,-1)--
++(1,0)-- ++(0,1)-- ++(1,0)-- ++(0,-1)-- ++(1,0)-- ++(0,-3)-- ++(-2,0)--
++(1,0)-- ++(0,-3)-- ++(1,0)-- ++(0,-1)-- ++(-6,0)-- ++(0,3)-- ++(2,0)--
++(0,-1)-- ++(1,0);
\end{tikzpicture}
```

Index

This index only contains automatically generated entries. A good index should also contain carefully selected keywords. This index is not a good index.

Data formats, *see* Formats

File, *see* Packages and files

Handlers for keys, *see* Key handlers

Layout, *see* Page layout

Node, *see* Predefined node

Options for graphics, *see* Graphic options and styles

Options for packages, *see* Package options

Styles for graphics, *see* Graphic options and styles