Ti*k*Z & PGF

Manual for Version 3.0.1a

```
\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]{
```

```
\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
        } } };
```

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

Till

The Ti*k*Z and PGF Packages

Manual for version 3.0.1a

`http://sourceforge.net/projects/pgf`

Till Tantau*

Institut für Theoretische Informatik
Universität zu Lübeck

2017 年 2 月 21 日

目录

*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.

1　介绍

欢迎阅读 Ti*k*Z 文档并运行 PGF 系统。它们源于一个用于生成图形的很小的 LaTeX 宏包，那时我 (Till Tantau) 还在博士论文中直接用 pdfLaTeX 生成图形，现在它已经成为一门功能丰富的图形语言，仅手册就已逾千页。Ti*k*Z 中的大量选项经常使初学者畏惧不前；但幸运的是，本文档采用大量渐进式的教程告诉你关于 Ti*k*Z 所应知道的一切，你甚至不必阅读其余的内容。

我希望从 "Ti*k*Z 是什么?" 开始。它基本上是一组画图的 TeX 命令。举例来说，代码`\tikz \draw (0pt,0pt) --(20pt,6pt);`得到直线 而代码`\tikz \fill[orange] (1ex,1ex)circle (1ex);` 则生成了 。某种意义上，你是在用 Ti*k*Z 编程生成图形，这和你使用 TeX 编程生成文档是一样的。这也是它名字的意义：Ti*k*Z 是 "Ti*k*Z ist *kein* Zeichenprogramm" 的迭代缩写，这也是沿袭了 "GNU is not unix" 的传统。它意味着 "Ti*k*Z 不是一个画图系统"，警示读者的误解。Ti*k*Z 使你可以充分利用 "TeX 的编译方式" 生成图形：快速生成简单的图形，准确的定位，宏的利用，还有超级好的排版。你同样还要遭遇所有 TeX 的缺点：陡峭的学习曲线，没有 "所见即所得"，微小的改动也需要长时间的再编译，还有代码并不显示它将呈现的样子。

现在我们了解了 Ti*k*Z 是什么，那么 "PGF" 又是什么呢? 如前所述，Ti*k*Z 作为一个项目致力于 TeX 图形命令宏的实现，这组宏可以应用于 pdfLaTeX 和其它典型的（基于 PostScript）LaTeX 驱动中。换言之，我想实现一个用于 TeX 的 "便携的图形格式"，这也是 PGF 的来历。这些早期的宏现在依然有效并构成了手册中所述系统的 "基本层"，但作者用于交互的主体却已经是 Ti*k*Z ，它本身即为一种完整的语言。

1.1　Ti*k*Z 的下层

在 Ti*k*Z 以下实际有两个层次：

系统层： 该层对 "驱动" 做了一个彻底的抽象。所谓驱动，就是一个类似`dvips`或 `dvipdfm`程序，它将输入的`.dvi`文件转为`.ps`或`.pdf`文件。（`pdftex`程序也一样称为驱动，即使它不将`.dvi`文件作为输入。别太介意哟。）每个驱动都有属于自己的生成图形的语法，这使每个想用一种可移植方式生成图形的人都头疼不已。PGF 的系统层将不同驱动的语法间的差别 "抽取了出去"。比如，系统命令`\pgfsys@lineto{10pt}{10pt}`将路径延伸到 `{pgfpicture}` 中的坐标 (10pt, 10pt)。根据具体用于处理文档的程序（驱动），如`dvips`、`dvipdfm`、或`pdftex`，系统命令将被转换为不同的`\special`命令。系统层尽可能是一位 "极简主义者"，因为每多一个额外的命令，在将 PGF 引入到新驱动中就会多做一些工作。

作为用户，你不会直接用到系统层。

基础层: 基础层提供了一整套基本命令，可以让你用更加简易的方式画出复杂的图形，而不是直接使用系统层命令。比如，系统层并未提供生成圆的命令，因为圆可以用更为基础的 Bézier 曲线生成。然而，作为用户，你需要有一个生成圆的简单命令（至少我是需要的），而不是必须写出半页的 Bézier 曲线坐标。这样，基础层就给出了一个命令`\pgfpathcircle`让你生成所需要的曲线坐标。

基础层包含一个*内核*，在内核中有几个相互依赖的宏包，这些宏包只能被整体加载，而其它模块则是采用特定目的的命令由内核扩展出来的，比如节点管理或绘制界面。举例来说，BEAMER 宏包只使用了内核，而没用引用比如`shapes`的其它模块。

理论上，Ti*k*Z 本身只是 "前端" 中的一种，所谓 "前端" 是若干命令或特殊语法的组合，它使得基础层用起来更加容易。直接使用基础层会使得代码过于冗长。举例来说，要画一个简单的三角形，使用基础层的话你得用 5 条命令：第一条是在三角形第一个角处开始画一条路径，然后向第二个角延伸，

再转向第三个，再闭合这条路径，最后才能实际绘制这个三角形（而非填充它）。而使用 Ti*k*Z 前端，这些烦杂动作将简化为类似于 METAFONT 的命令：

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

Ti*k*Z 其实是 PGF 唯一 “真正的” 前端。它可以使你使用 PGF 所有功能，却非常易用。它的语法是 METAFONT 和 PSTRICKS 的混合体，其中还有我自己的一些想法。在 Ti*k*Z 之外还有些前端，但更类似于 “技术研究”，而不是有意作为 Ti*k*Z 的替代品。pgfpic2e 前端重新实现了标准 LaTeX {picture} 环境和类似 PGF 基础层中\line和 \vector的命令。这一层并非真正 “必需”，因为至少pict2e.sty 宏包可以和重实现的 {picture} 环境同样出色地完成工作。该宏包其实是想证明实现前端是一件很容易的事。

由于大多数用户只使用 Ti*k*Z 而没人直接用系统层，本手册首先介绍 Ti*k*Z ；基础层和系统层在后面再作介绍。

1.2 同其它图形宏包比较

Ti*k*Z is not the only graphics package for TeX. In the following, I try to give a reasonably fair comparison of Ti*k*Z and other packages.

Ti*k*Z 不是 TeX 中惟一的图形宏包。下面我试着对 Ti*k*Z 和其它宏包作一个理性公平的比较。

1. The standard LaTeX {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 LaTeX'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 Ti*k*Z, 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 Ti*k*Z syntax is more consistent than the pstricks syntax as Ti*k*Z 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 Ti*k*Z, it is very small, which may or may not be an advantage.

5. The metapost program is a powerful alternative to Ti*k*Z. It used to be an external program, which entailed a bunch of problems, but in LuaTeX it is now build 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 Ti*k*Z for users who do not wish to “program” their graphics as is necessary with Ti*k*Z and the other packages above. There is a conversion program that will convert xfig graphics to Ti*k*Z.

1.3 工具包

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 TeX 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 如何阅读本手册

This manual describes both the design of Ti*k*Z 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 Ti*k*Z, 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 Ti*k*Z. 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 作者和致谢

PGF 系统的主体和文档由 Till Tantau 创作。团队的另一名成员是 Mark Wibrow，它负责的工作包括，PGF 的数学引擎、许多形状、装饰引擎、和矩阵。第三位成员是 Christian Feuersänger，他贡献了浮点库、图形外部化、扩展键处理、和手册中的自动链接。

其它贡献来自于 Christophe Jorssen、Jin-Hwan Cho、Olivier Binda、Matthias Schulz、Renée Ahrens、Stephan Schuster、和 Thomas Neumann。

另外，很多人通过写信、发现 bugs、或发送库和补丁等对 PGF 系统做出了贡献。感谢他们所有人，因为人员很多，这里不一一列出。

1.6 获取帮助

When you need help with PGF and Ti*k*Z, please do the following:
当你使用 PGF 和 Ti*k*Z 需要帮助时，请尝试如下方法：

1. 阅读本手册，至少阅读问题涉及的部分。
2. 如未能解决问题，就去看看 PGF 和 Ti*k*Z 的开发页（参见本文档标题）。也许有人已经报告了同样的问题，而某些人已经找到了解决办法。
3. 在网上你可以找到很多论坛，从中获得帮助。在那里你可以获得帮助，报告 bug，加入邮件列表，等等。
4. 在你提交一个 bug 报告特别是一个关于安装的 bug 报告之前，请确认确实存在这个 bug。尤其要看看在 TEX 编译文件时`.log`文件中的结果。`.log`文件会表明所有在正确目录中加载的正确文件。几乎所有的安装问题都可以通过观察 `.log`文件得以解决。
5. *最后一招*就是你给我写信了 (Till Tantau)，或者关于数据引擎的问题写给 Mark Wibrow。因为数量太多，我其实不太关注收到的邮件。我因此不能保证你的邮箱会被及时回复，有可能根本得不到回复。如果你写信发给 PGF 邮件列表中，解决问题的机会将会多少增加一些。（一般我在有时间时会阅读列表并回答问题。）