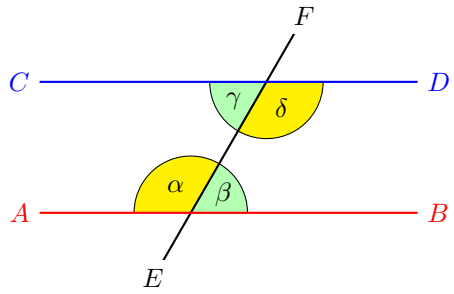
Part III

TikZ ist kein Zeichenprogramm

by Till Tantau



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

```
\begin{tikzpicture}
  \draw[fill=yellow] (0,0) -- (60:.75cm) arc (60:180:.75cm);
  \draw(120:0.4cm) node {$\alpha$};

  \draw[fill=green!30] (0,0) -- (right:.75cm) arc (0:60:.75cm);
  \draw(30:0.5cm) node {$\beta$};

  \begin{scope}[shift={(60:2cm)}]
    \draw[fill=green!30] (0,0) -- (180:.75cm) arc (180:240:.75cm);
    \draw (30:-0.5cm) node {$\gamma$};

    \draw[fill=yellow] (0,0) -- (240:.75cm) arc (240:360:.75cm);
    \draw (-60:0.4cm) node {$\delta$};
  \end{scope}

  \begin{scope}[thick]
    \draw  (60:-1cm) node[fill=white] {$E$} -- (60:3cm) node[fill=white] {$F$};
    \draw[red]                    (-2,0) node[left] {$A$} -- (3,0) node[right]{$B$};
    \draw[blue,shift={(60:2cm)}] (-3,0) node[left] {$C$} -- (2,0) node[right]{$D$};

    \draw[shift={(60:1cm)},xshift=4cm]
    node [right,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., ${\color{red}AB} \mathbin{\|} \color{blue}CD$,
      then $\alpha = \delta$ and $\beta = \gamma$.
    };
  \end{scope}
\end{tikzpicture}
```

8 设计原理

TikZ 的意思是“TikZ ist kein Zeichenprogramm”，意思是 TikZ 并非一个用户图形化交互式绘图程序，这一节主要讲述 TikZ 幕后的设计原理。

要使用 TikZ，对于 LaTeX 用户需要在导言区中 `\usepackage{tikz}`，对于 plain TeX用户需要 `\input tikz.tex`。TikZ 的主要目标是提供一种易于学习且易于使用的绘图语法，让你的绘图任务更轻松。

TikZ 的命令和语法受许多软件包的影响，它的基本命令名与路径操作概念皆来源于 METAFONT，选项机制模仿 PSTRICKS，风格的概念取材于 SVG。为了让这些元素可以交融起来，就需要一些折中的方案。我也添加了一些我自己的思想，比如 一些箭头风格和坐标变换。

以下是 TikZ 内在的基本设计原理梗概：

1. 用于设定点的特定语法
2. 用于设定路径的特定语法
3. 对路径的操作
4. 图形参数的键值语法
5. 用于设定结点的语法
6. 用于设定树形结构的语法
7. 图形参数的组合
8. 系统坐标变换

8.1 用于设定点的语法

TikZ 提供了一套专用于设定点及其坐标的语法，最简单情况下，你只需提供以逗号间隔并以圆括号囊括的 2 个 TeX 维度坐标值即可，比如 (1cm, 2pt)。

你也可以使用极坐标模式指定点，只需提供两个以冒号为间隔并以圆括号囊括的两个极坐标值，比如 (30:1cm)，意思是点在“30 度方向上 1cm 处”。

如果在 PGF 的 xyz 坐标系中设定点坐标时，未给出相应单位，譬如 (2,1)，那么默认单位是 `cm`。

你也可以使用类似于 (1,1,1) 这样的坐标形式在 xyz 坐标系中设定点。也可以基于已有结点的锚点来设定点，譬如 (first node.south)。

若是在点的坐标前添加两个 “+” 作为前缀，譬如 ++(1cm, 0pt)，这表示在最后设定的点位置的基础上沿 x 方向推进 1cm 所得到的新点位置，利用这个新的点位置来设定点。使用这种设定点的方法就可以实现相对运动。比如 (1,0) ++(1,0) ++(0,1) 表示设定了 3 个点，坐标分别为 (1,0) (2,0) (2,1)。

也可以使用一个 “+” 作为点坐标前缀，这也是一种相对位置指定模式，但它不改变当前点位置。比如，(1,0) +(1,0) +(0,1) 表示的三个点分别为 (1,0) (2,0) (1,1)，从中即可看出 “++” 与 “+” 的区别。

8.2 用户设定路径的语法

TikZ 绘图实质上就是在绘制路径。一条路径是由一系列直线或曲线构成的，它们之间不一定是相互衔接的。TikZ 设定路径很容易，部分语法与 METAPOST 很相像，比如要设定一个三角形的路径，可：

```
(5pt,0pt) -- (0pt,0pt) -- (0pt,5pt) -- cycle
```

当你使用 `\draw` 命令绘制这条路径时，就可以得到 ◺。

8.3 对路径的操作

一条路径虽然有一系列直线或曲线构成，但它并没有设定在这条路径上会发生什么。你可以绘制一条路径，填充一条路径，渲染一条路径，裁剪一条路径，或者将这些操作组合起来。所谓绘制一条路径可以被认为是拿着一只某种型号的画笔沿着所设定的路径移动，这样就可以勾勒出图案。填充一条路径就是以均匀的颜色填充在一条封闭的路径内部，如果所设定的路径不是封闭的，TikZ 会自动将其封闭。

设定一条路径 `\path (0,0) rectangle (2ex,1ex);`，然后设定 `\path` 命令的 draw 选项 `\path[draw] (0,0) rectangle (2ex,1ex);`，就可以绘制一个矩形。对于 `\path[draw]` 可使用其缩写形式 `\draw`。

要填充一条路径可以使用 `\path[fill]`，我们通常使用其简写形式 `\fill`。也可以使用命令 `\path[draw,fill]` 同时绘制路径并填充，其简写形式为 `\filldraw`。

渲染也是通过指定 `\path` 的一个选项 shade 来实现的其简写形式为 `\shade`，另外也可以与 `\draw` 命令搭配使用，即 `\shadedraw`。

裁剪是用过指定 `\path` 的选项 clip 来实现的，命令可简写为 `\clip`，但是它与 `\draw` 命令配合使用时，没有 `\drawclip` 形式，而是 `\path[draw,clip]` 或 `draw[clip]`。

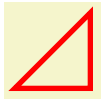
以上所有命令都可以在 tikzpicture 环境中使用。

TikZ 允许你使用不同的颜色进行 draw、fill、shade。

8.4 图形参数的键值语法

TikZ 在绘制或填充路径时，很多图形参数可以改变图形渲染效果，譬如调配颜色、虚线模式、裁剪区域、线宽等参数。在 TikZ 中，这些参数是使用一种键值对的形式进行设定，比如 color=red，这些键值对作为可选参数传递给 `\draw` 或 `\fill` 等绘图命令。这种方式与 PSTRICKS 类似。

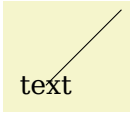
下面给出一个示例，绘制一个红色、粗线型的三角形：



```
\tikz \draw[line width=2pt,color=red] (1,0) -- (0,0) -- (1,1) -- cycle;
```

8.5 用于设定结点的语法

TikZ引入了一种用于向图形中添加文本、结点的语法。当你设定一条路径时，可以像下面这样添加结点：




```
\tikz \draw (1,1) node {text} -- (2,2);
```

结点被插入到路径的当前位置，但是它是在路径被渲染完毕后才被绘制。当给定一些特定选项时，比如 `\draw (1,1) node[circle,draw] {text}`并非仅仅是在当前位置插入“text”，而是让 “text” 由一个圆形包围并画出这个圆。

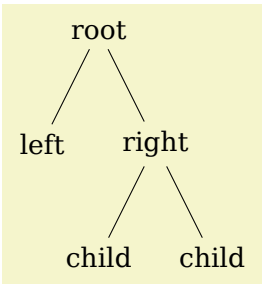
你可以使用 `name=`⟨node name⟩ 对结点进行命名，以便于在别处引用所命名的结点，也可以直接将结点名置入圆括号中，并将其作为结点的外部文本，比如 `node[circle] (name) {text}`。

结点有一些已定义好的形状，比如矩形、圆形、椭圆等，自定义结点形状也是可以的。


```
\begin{tikzpicture}
\draw (1,1) node [circle,draw] (n1) {text} -- (2,2);
\end{tikzpicture}
```

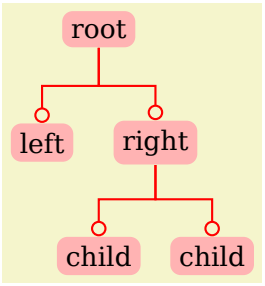
8.6 用于设定树形结构的语法

作为结点语法的补充，TikZ引入了一种专用于绘制树形结构的语法，它可以与结点设定语法混合使用，仅仅需要再多记忆几条新命令。实际上，一个结点后面可以连接任意数量的子结点，每个子结点都有关键词 `child` 指定。子结点本身也是结点，它们的每一个都可以再跟随许多子结点。

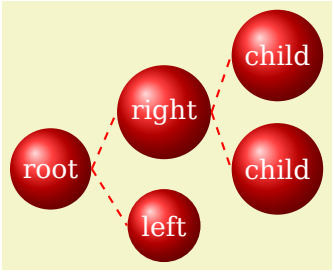



```
\begin{tikzpicture}
  \node {root}
    child {node {left}}
    child {node {right}
      child {node {child}}
      child {node {child}}
    };
\end{tikzpicture}
```

由于树是由结点构成的，因此可以使用一些选项来调整树的形状。对于上面示例的树，下面采用不同的选项进行绘制：




```
\begin{tikzpicture}[edge from parent fork down]
  \tikzstyle{every node}=[fill=red!30,rounded corners]
  \tikzstyle{edge from parent}=[red,-o,thick,draw]
  \node {root}
      child {node {left}}
      child {node {right}
        child {node {child}}
        child {node {child}}
      };
\end{tikzpicture}
```




```
\begin{tikzpicture}
    [parent anchor=east,child anchor=west,grow=east]
  \tikzstyle{every node}=[ball color=red,circle,text=white]
  \tikzstyle{edge from parent}=[draw,dashed,thick,red]
  \node {root}
      child {node {left}}
      child {node {right}
        child {node {child}}
        child {node {child}}
      };
\end{tikzpicture}
```

8.7 图形参数的分组

一组图形参数通常应当用来作用于有限条路径的绘制与填充。譬如，要绘制一组宽度均为 1pt 的直线，但是这组直线之外的其他路径就没有这种要求。我们可以通过设定域 (scope) 环境来实现对图形参数的作

用范围约束。另外，在域环境之内中所设定的参数可以覆盖域环境外的所设定的图形参数，单条绘图命令也可以认为是一个很小的域环境。在下面的示例中，使用域环境与单条绘图命令中的图形参数覆盖模式绘制了 3 条红线，2 条绿线和 1 条蓝线。

```
\begin{tikzpicture}
  \begin{scope}[color=red]
    \draw (0mm,10mm) -- (10mm,10mm);
    \draw (0mm, 8mm) -- (10mm, 8mm);
    \draw (0mm, 6mm) -- (10mm, 6mm);
  \end{scope}
  \begin{scope}[color=green]
    \draw              (0mm, 4mm) -- (10mm, 4mm);
    \draw              (0mm, 2mm) -- (10mm, 2mm);
    \draw[color=blue] (0mm, 0mm) -- (10mm, 0mm);
  \end{scope}
\end{tikzpicture}
```

tikzpicture环境本身也可以视为是一个 scope 环境，也就是说也可以设置这个环境的图形参数作用于 tikzpicture 环境内的所有绘图命令。

8.8 坐标变换系统

TikZ完全地借助于 pgf的坐标变换系统来实现左标变换。pgf也支持画布 (canvas) 变换 —— 一种更低级的坐标变换系统，但这个变换系统不能由 TikZ直接调用，因为：画布变换难以使用，容易破坏图形结构；使用画布变换后，PGF 就难以对结点及其轮廓进行定位了。一句话来区分一下坐标变换与画布变换：坐标变换不改变线宽与文字大小、形状，而画布变换让所有的图形都发生变形。