tikz 制图指南

在 xelatex 指南之上

主要是文档中 *inline* 的制图，大型复杂制图还是考虑外部绘图软件。

万泽[1] | 德山书生[2]

版本：0.01

[1]作者：湖南常德人氏
[2]编者：德山书生，湖南常德人氏。邮箱：a358003542@gmail.com。

前言

TikZ is not an interactive drawing program.

1.Graphics with TikZ Andrew Mertz and William Slough

2.A very minimal introduction to TikZ Jacques Crémer

3.the tikz 官方文档，这个用 texdoc 命令调不出官方文档，用 google 搜索“tikz pdf”吧

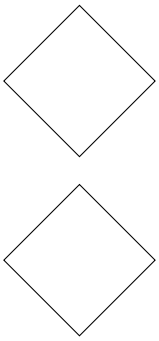
目 录

准备工作

tikz 宏包的加载是必须的，因为我这里 myconfig.sty 里面的已经加载了 chemfig 宏包，而 chemfig 宏包又加载了 tikz 宏包，所以不需要修改什么了。

```
\tikz{\draw (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;}

\begin{tikzpicture}
\draw (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;
\end{tikzpicture}
```



有两种使用方法，一种命令式的，一种环境式的。命令式用 tikz 命令包围起来，命令式是 inline 模式的。环境式用 tikzpicture 命令包围起来。

这里推荐 KTikZ 软件，在 ubuntu 软件中心里面就有，用这个软件可以边写 tikz 代码边看画的图形，还可以导出图形，早期推荐先使用这个软件练练手熟悉下 tikz 制图代码。值得一提的是这个软件不支持中文注释，会生成乱码。

tikz 基础

tikz 系统简介

一般使用 frontend layer（前端）就够了，有时可能需要用到 basic layer（基本层）的命令。基本层是架构在系统层上的，里面是画图的最底层的驱动，绝大多数用户完全不需要了解这些。

tikz 的画图命令要某在`\tikz{}` 里面，要某在 tikzpicture 环境里面。

单位

默认的长度单位是 cm。

第一个例子

画网格

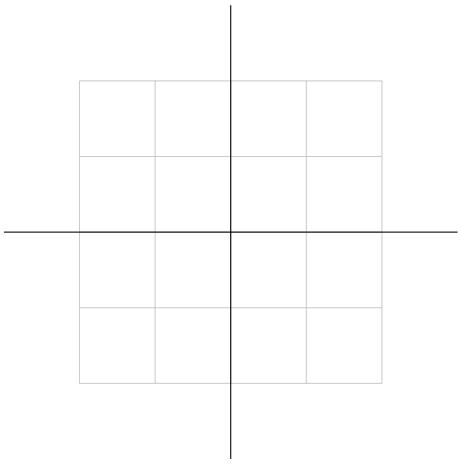
```
\begin{tikzpicture}
\draw[step=1,color=gray!40] (-2,-2) grid (2,2);
```

```
\end{tikzpicture}
```


step 是网格之间的间距，color 是网格的颜色。第一个坐标点是网格的左底点，第二个坐标点是网格的右顶点。我们可以看到 tikzpicture 下每一条命令最后都要跟一个分号; 。

画直线

```
\begin{tikzpicture}
\draw[step=1,color=gray!40] (-2,-2) grid (2,2);
\draw (-3,0) -- (3,0);
\draw (0,-3) -- (0,3);
\end{tikzpicture}
```



画直线就是两个坐标点相连，中间 -- 符号表示直线的意思。之前网格是 grid 表示网格的意思。

如果几个点用 -- 符号连接起来，表示这几个点连着来画几条折线，有多个画直线命令依次执行的意思。

直线带上箭头

draw 命令可以跟上可选项 **->**，这样直线的右端就有一个箭头了。此外还有：**-»**，**->|**，**-to**，**-latex**，**-stealth**。

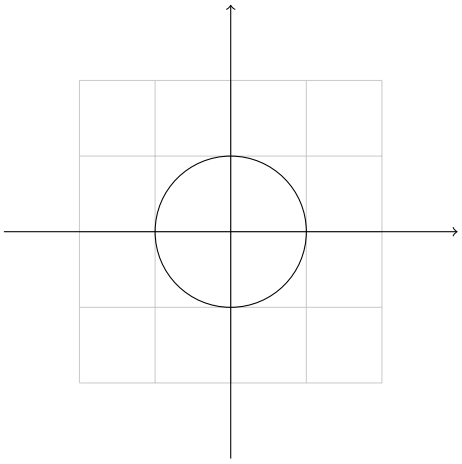
他们的效果从上到下依次演示如下：

类似的还有左端比如 **<-**，或者两端比如 **latex-latex**，这里就不多说了。

画圆

接著上面的图案画一个圆，加入了以下代码：

```
\draw (0,0) circle (1);
```

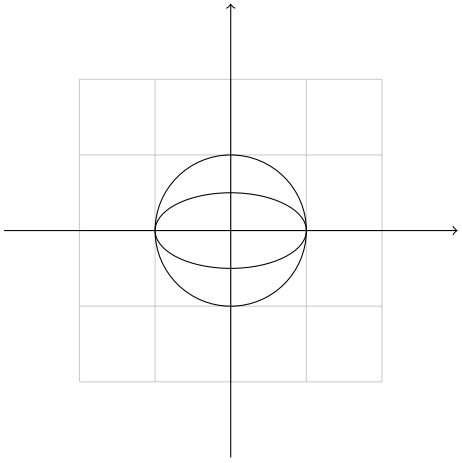


其中第一个点是圆中心，circle 表示画圆，第二个参数是半径大小。

画椭圆

接著画一个椭圆：

```
\draw (0,0) ellipse (1 and 0.5);
```



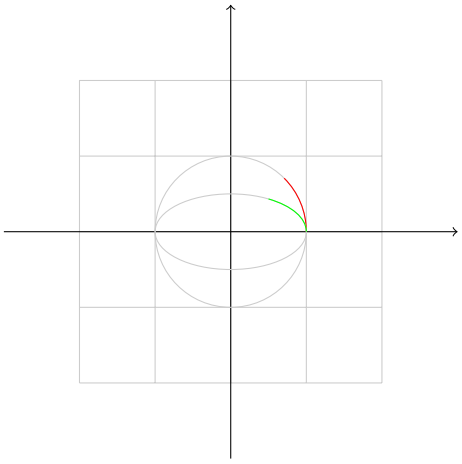
这里第一个点是椭圆的中心点，ellipse 表示画椭圆，后面参数两个值第一个是 a 也就是椭圆的半长轴，第二个是 b 也就是椭圆的半短轴。

画弧线

```
1 \begin{tikzpicture}
2 \draw[step=1,color=gray!40] (-2,-2) grid (2,2);
3 \draw[->] (-3,0) -- (3,0);
4 \draw[->] (0,-3) -- (0,3);
5 \draw[color=gray!40] (0,0) circle (1); %
6 \draw[color=red] (1,0) arc (0:45:1);
7 \draw[color=gray!40] (0,0) ellipse (1 and 0.5);
8 \draw[color=green] (1,0) arc (0:60:1 and 0.5);
9 \end{tikzpicture}
```

最基本的画弧线的命令如上代码第 5 行，其中第一个点是弧线的起点，然后 arc 表示画弧线，接下来括号里面的三个参数：第一个参数是开始的角度，第二个参数是结束时的角度，第三个参数是弧线对应圆的半径。对比第 4 行画的浅灰色的圆可以看出他们之间的关系。

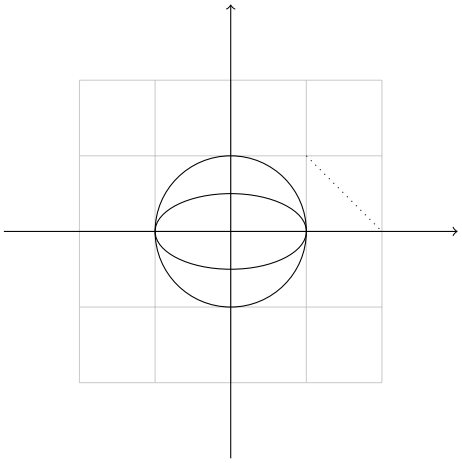
上面代码第 7 行画弧线增加了一个 and 和一个参数，这个时候画的弧线是根据椭圆来的，其中 1 是椭圆的半长轴，0.5 是椭圆的半短轴。对比第 6 行画的浅灰色的椭圆可以看出他们的关系。



点的定义

tikz 中定义一个点方便之后使用：

```
1 \path (1,1) coordinate (point001);
2 \path (2,0) coordinate (point002);
3 \draw[dotted] (point001) -- (point002)  ;
```

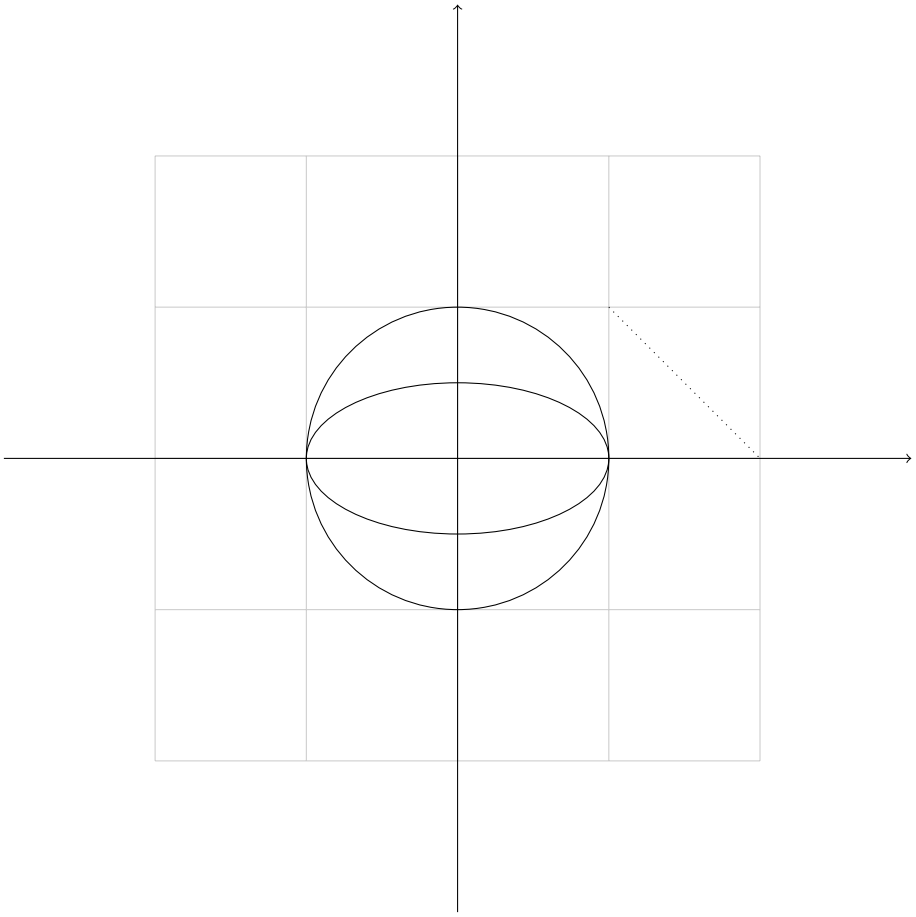


这里代码的第 6，7 行定义了两个点，名字叫做 point001 和 point002。然后用这两个点作为参数画了一个直线，这个直线有可

选项 **dotted**，点线样式。

放大图形

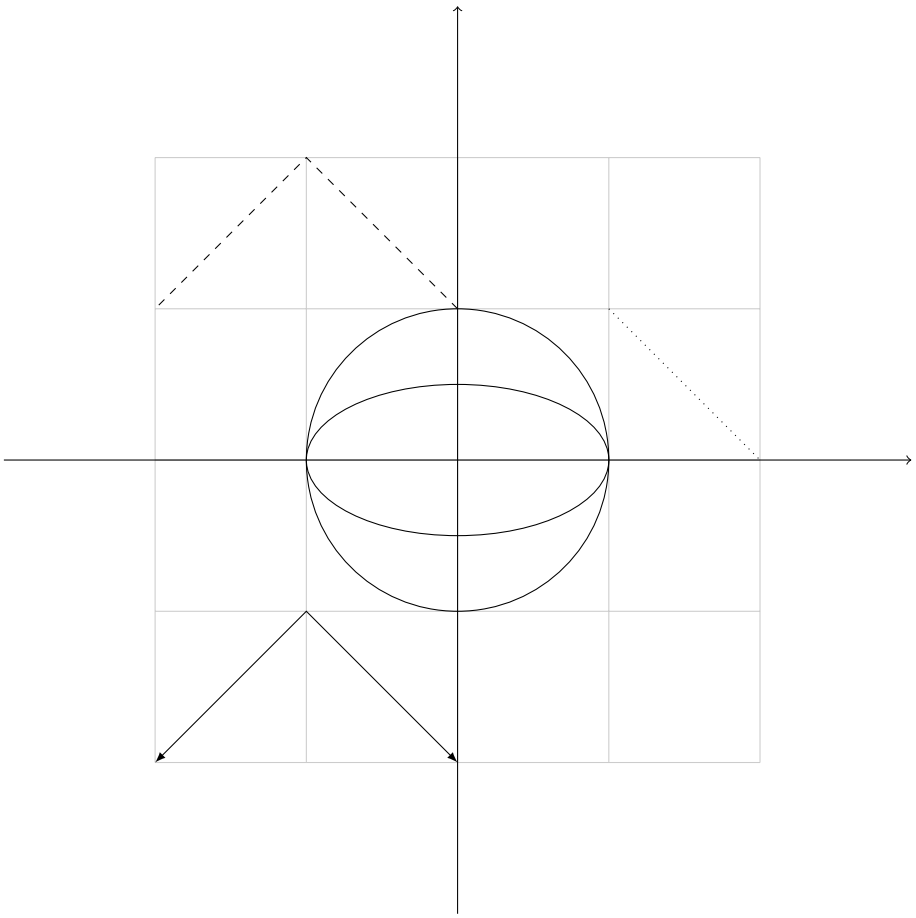
在 tikzpicture 环境后面跟上可选项 [scale=2]，即将图形放大两倍。注意控制别越界了。



点的相对偏移

现在加上这样两行代码：

```
1 \draw[<->] (0,-2) -- ++(-1,1) -- ++(-1,-1);
2 \draw[dashed] (0,1) -- +(-1,1) -- +(-2,0);
```

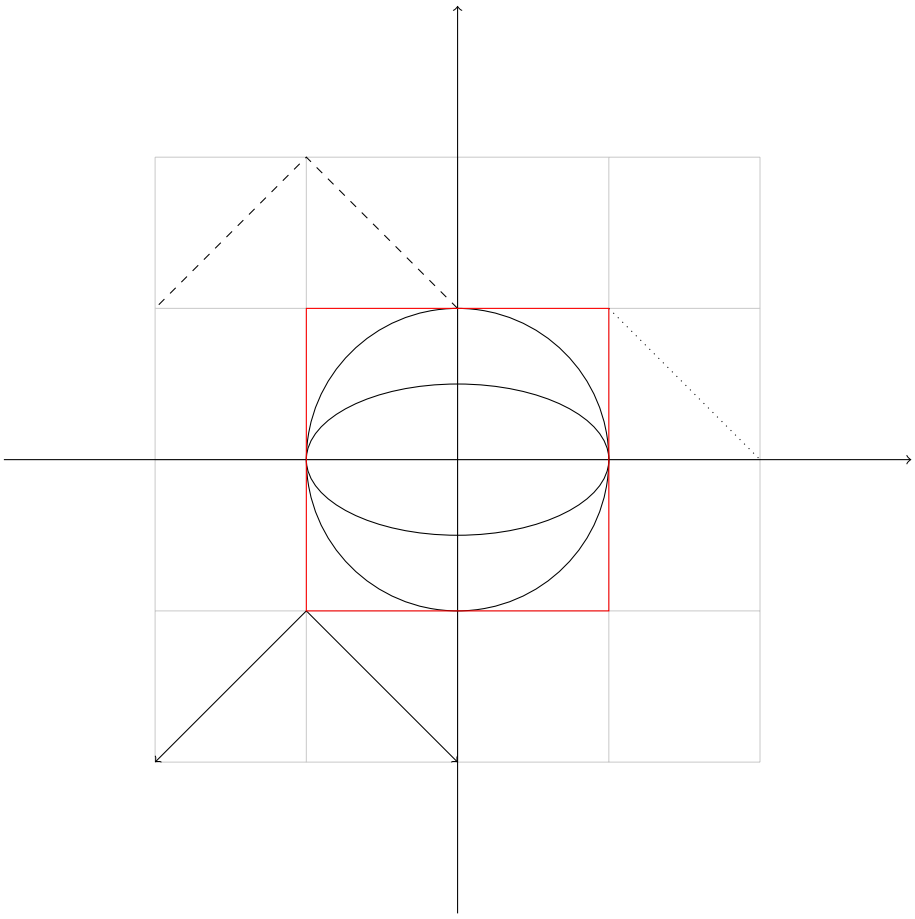


tikz 中有一个重要的概念，当前点，然后点可以通过当前点根据相对偏移来确定一个新的点。上面代码第 9 行的 **++** 符号和第 10 行的 **+** 符号都根据当前点然后进行了 Δx 和 Δy 的相对偏移从而确定了一个新的点。这两个符号的区别在于是不是更新当前点数据。++ 符号更新当前点，而 + 符号不更新。

画长方形

现在加上这一行代码来画一个长方形：

```
\draw[color=red] (-1,-1) rectangle (1,1);
```



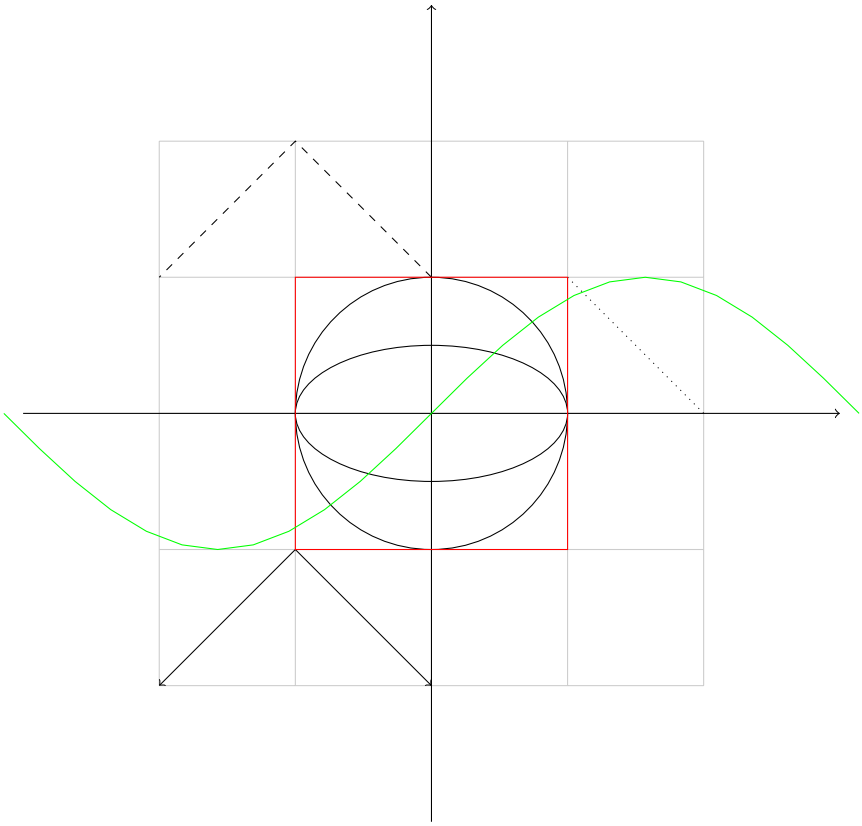
这里使用了可选项 **color=red** 来控制线条的颜色，然后画长方形的第一个点是左底点，rectangle 表示画长方形，第二个点表示右顶点。

画函数

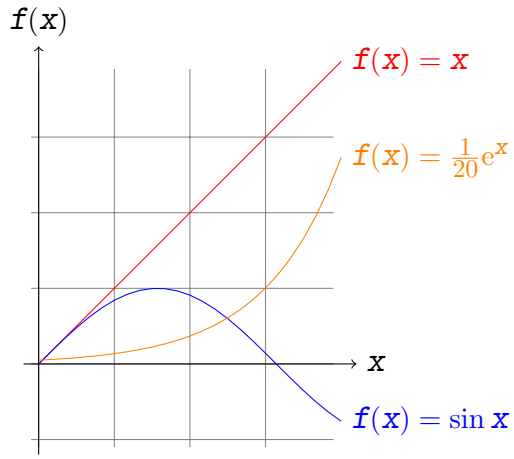
画函数的功能是通过外部程序 gnuplot 来实现了，所以需要打开`--shell-escape`，或者`--enable-write18`

这里最后加了一个语句：

```
\draw[domain=-pi:pi,color=green] plot function{sin(x)};
```



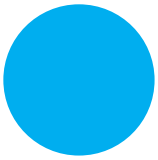
这里可选项 **domain=-pi:pi** 控制画的函数的 x 范围，可以直接用 pi 表示 π，然后接下来 plot function 表示画一个函数，接下来的花括号里面放着 gnuplot 的各种命令，这里就是简单的 $sin(x)$



颜色填充

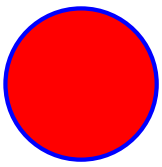
fill 命令就是填充某种颜色的形状，后面跟个 **color** 可选项设置填充的颜色，默认是黑色。比如画一个填充颜色的圆：

```
\fill[cyan] (0,0) circle  (1) ;
```



为了简单起见，draw 命令可以加上 fill 可选项，然后和上面类似的有：

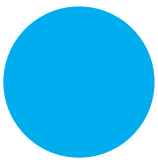
```
1 \begin{tikzpicture}
2 \draw [color=blue,fill=red,ultra thick,] (0,0) circle (1);
3 \end{tikzpicture}
```


注意到线条的颜色和填充颜色的控制。

填充没有线条

如果你不希望有线条，那么使用 path 命令可以做到这点。

```
\begin{tikzpicture}
\path[fill=cyan] (0,0) circle  (1) ;
\end{tikzpicture}
```



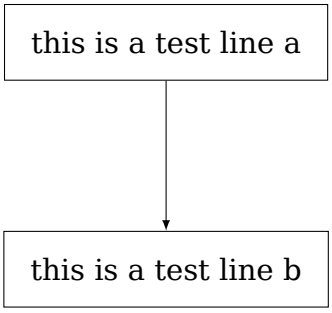
node 命令

node 命令主要用于插入文本，不过最好将其理解为接口。XeLaTeX 文档内部各个命令等都可以使用，然后外面包围一个形状，如 rectangle 长方形，circle 圆等。

```
\newcommand{\testlinea}{this is a test line a}
\newcommand{\testlineb}{this is a test line b}
\begin{tikzpicture}
%\fill[cyan] (0,0) circle  (1) ;
```

```
\node[shape=rectangle,draw,inner sep=10pt] at (0,0) (a) {\testlinea};
\node[shape=rectangle,draw,inner sep=10pt] at (0,-3) (b) {\testlineb};
\draw[-latex](a) -- (b);
\end{tikzpicture}
```

这里我们看到 LaTeX 里面自定义的命令是可以正常使用的，然后可选项 **shape** 的意思是外面包围的形状是长方形，**draw** 就是画这个形状是用的 draw 命令方法，比如 fill 就会填充。**inner sep** 控制外面的形状和内部文本之间的间距。然后 **at (0,0)** 控制整个图形的位置，然后 **(a)** 表示整个图形的名字，后面可以调用的，可以看作接口把。然后后面就是 LaTeX 的内容了。



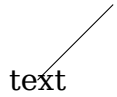
插入文本的位置

node 命令的可选项 **left**，**right**，**above**，**below** 用于控制插入文本的位置。此外还有 **above right**，**below right**，**above left** ，**below left** 等。

文本对齐控制

用 **align=left**，**align=right**，**align=center** 来控制。

在画图形的时候插入文本



在画图形的时候某个当前点下可以直接 node 接某个文本。

clip 命令

clip 就是剪切的意思，就是通过 clip 命令按照某个形状来剪切，外面的图形都不保留，可以跟一个可选项 **draw**，这样剪切的时候同时画出了这个形状。

抛物线

path 路径闭合

任何构建的 path 最后都可以通过 **-- circle** 将其闭合起来。

filldraw 命令

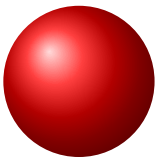
是 draw 命令和 fill 命令的结合。**fill=** 可选项调整填充的颜色，**draw=** 可选项调整画的线条的颜色。

阴影

shade 命令和 fill 命令的区别就是填充的颜色是渐变的，其他类似。

其可选项有 **top color 和 bottom color** 表示上下渐变的颜色，**left color 和 right color**，**innercolor 和 outercolor**，**ball color** 让颜色渐变像一个有光照的球。

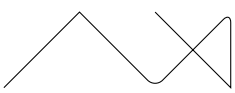
小红球



scope 环境

scope 环境就是作用域控制，一个局域环境，参数只影响内部。

更灵活的 scope 控制



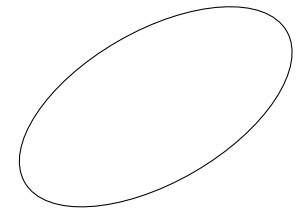
变形

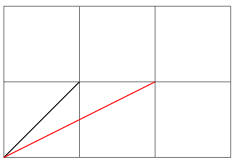
xshift x 坐标轴平移 yshift y 坐标轴平移 rotate 旋转。注意 xshift 默认的单位并不是 cm，如果要单位是 cm 需要写出来。

旋转图形

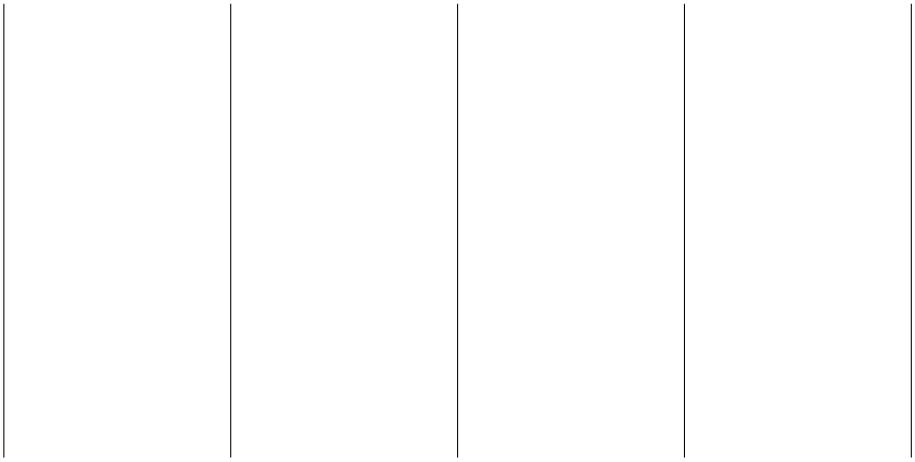
后面加上可选项 **rotate=30** 即可，意思是图形逆时针旋转 30 度。

```
1 \begin{tikzpicture}
2 \draw (0,0)[rotate=30]  ellipse (2 and 1);
3 \end{tikzpicture}
```





迭代语句



其中... 表示一直这样有规律下去生成迭代列表。

文本中的图片

wherever you want

baseline 选项

这个主要控制 inline 模式下图片的位置，默认 **baseline=0pt**。

样式

style，特定图形的样式。定义一个样式比如 style001 如下：

```
style001/.style={color=red,fill=red!20}
```

原有样式修改

```
help lines/.append style=blue!50
```

附加之后最新的样式胜出。

样式带参数

red

blue

样式参数有默认值

default

blue

定义点

定义绝对点

```
\path (0,29) coordinate (top-left);
```

path 命令后面跟着坐标点，然后 coordinate 后面跟着这个点的名字。这里规范为 coordinate 命令后面跟着就是点的名字，node 命令后面跟着 node 的名字。

定义相对点

```
1 \path (top-left) ++(1,-2) coordinate (name-point);
```

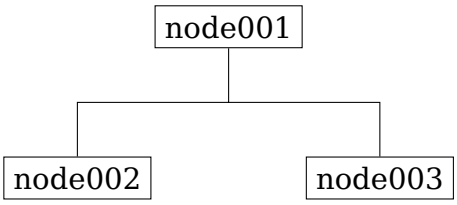
极座标

tikz 中的点也支持极座标表示，(30:1cm)，第一个参数是极座标里面的角度，第二个参数是半径。

node 命令中点的定义

从这里可以看到只要写上 draw 选项外面就会加上一个长方形，也就是 shape 的默认选项是 rectangle。如果你不希望外面有长方形，不写 draw 选项即可。

这里通过 node 命令定义了一个点，node001，在 (0,2) 那里。后面是可以使用的。



这里通过 **node cs:name=node003** 来获取之前那个 node 所在的点，然后通过 **anchor=north** 来定义那个 node 的接口在北边。除此之外的选项还有：**south**，**east**，**west**。这里 **|-**似乎是画垂直拐线的意思。上面的语法简写为可以 node002.north。

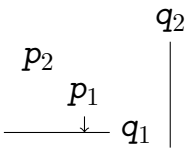
此外还有 **angle** 选项控制 node 接口的开口角度。

两个点定义出一个点

```
\begin{tikzpicture}
\node (p1) at (30:1) {$p_1$} ;
\node (p2) at (75:1) {$p_2$} ;
\draw (-0.2,0) -- (1.2,0) node[right] (xline) {$q_1$};
\draw (2,-0.2) -- (2,1.2) node[above] (yline) {$q_2$};

\draw[->] (p1) -- (p1 |- xline);
\end{tikzpicture}
```

这种形式 (p1 |- xline) 表示取第一个点的 x 和第二个点的 y 组成一个新的点。如果是 (p1 -| xline) 表示取第二个点的 x 和第一个点的 y 组成一个新的点。



两个 path 的交点

点的运算

在进行下面说的数学运算之前需要加载 calc 宏包：

```
\usetikzlibrary{calc}
```

基本格式是：

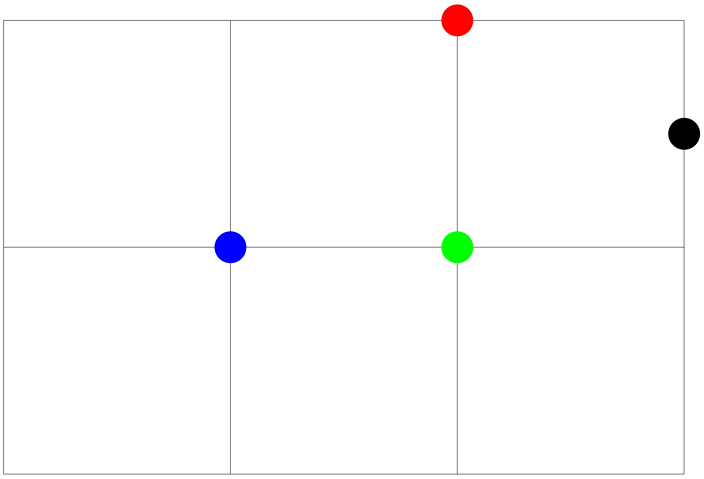
```
([options]$(一些运算)$)
```

这里$$ 表示这里有一些数学运算。里面的基本格式如下：

<factor>*<点><其他修饰>

```
\begin{tikzpicture}[scale=3]
\draw [help lines] (0,0) grid (3,2);
\fill [red] ($2*(1,1)$) circle (2pt);
\fill [green] (${1+1}*(1,0.5)$) circle (2pt);
\fill [blue] ($cos(0)*sin(90)*(1,1)$) circle (2pt);
\fill [black] (${3*(4-3)}*(1,0.5)$) circle (2pt);
\end{tikzpicture}
```

第一个红点是点 (1,1)，然后 x 和 y 都乘以 2 从而得到新点。后面情况类似，不同的是前面的乘法还可以加入更多的运算。



path 命令

path 路径是最基本的命令，draw 命令等价于\path[draw]，fill 命令等价于\path[fill]，filldraw 命令等价于\path[draw,fill]，

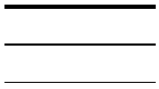
其他 clip，shade 命令情况类似。

path 之线条

虚线和点线

线条除了之前说的 dashed 和 dotted 两种样式之外，还有 loosely dashed，densely dashed 和 loosely dotted，densely dotted。比如：　　，这是 dashed 的三种，下面是 dotted 的三种：　　。

线条的粗细



其他选项还有 **ultra thin**, **very thin**, **thin**, **semithick**, **thick**, **very thick** and **ultra thick** 还有 **help lines** 选项那种很淡灰的样式。

或者直接通过可选项 line width 来定义。

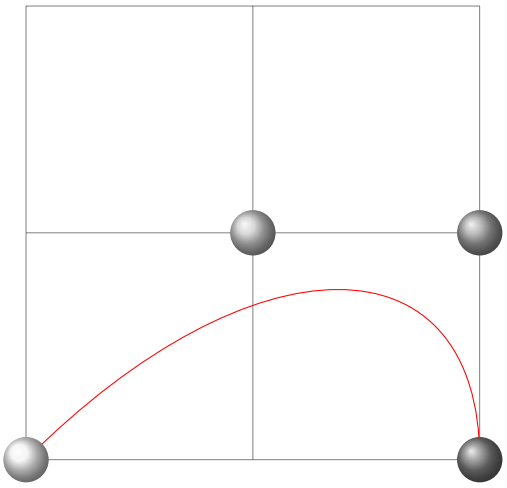
圆圆的拐角

贝塞尔曲线

贝塞尔曲线是四个点画出一个曲线，具体我现在还不太清楚。其中第一个点是起点，第四个点终点，然后另外两个点是控制点。

```
1 \begin{tikzpicture}[scale=3]
2 \draw[help lines] (0,0) grid (2,2);
3 \draw[color=red] (0,0) .. controls (1,1) and (2,1) .. (2,0);
4 \shade[ball color=gray!10] (0,0) circle (0.1);
5 \shade[ball color=gray!40] (1,1) circle (0.1);
6 \shade[ball color=gray!70] (2,1) circle (0.1);
7 \shade[ball color=gray] (2,0) circle (0.1);
8 \end{tikzpicture}
```

上面第 2 行代码就是画贝塞尔曲线的代码。



tikz 的一些例子