

Pionpill's TikZ Notebook

Pionpill ¹

学习 TikZ 的笔记，正在学习中，有待更新
本文主要根据 TikZ 官方手册 ² 学习

2021 年 9 月 20 日

¹笔名：北岸，电子邮件：673486387@qq.com，Github： <https://github.com/Pionpill>

²TikZ Github： <https://github.com/pgf-tikz/pgf>

目录

I 案例

一、三角函数：A Picture for Karl's Students	1
1.1 效果预览	1
1.2 基本曲线	2
1.2.1 直线	2
1.2.2 曲线	2
1.2.3 椭圆	3
1.2.4 矩形	3
1.3 网格	4
1.3.1 网格的绘制	4
1.3.2 样式设置	4
1.4 弧度	5
1.5 截图	5
1.6 抛物线与三角函数	6
1.6.1 抛物线	6
1.6.2 三角函数	6
1.7 填充与边线	7
1.7.1 基本填充与边线	7
1.7.2 渐变	8
1.8 指定坐标	8
1.9 线段与箭头	9
1.9.1 箭头样式	9
1.9.2 范围 scope	10
1.10 位移	11
1.11 循环	11
1.11.1 数字循环	11
1.12 增加文字	12
1.12.1 node 关键字	12
1.12.2 线段文字	13
1.13 最终结果	14
1.14 其他技巧	14

二、流程图: Diagrams as Simple Graphs	16
2.1 效果预览	16
2.2 节点设置	16
2.2.1 节点样式	16
2.2.2 节点位置	17
2.2.3 节点连线	18
2.3 矩阵布局	19
2.4 连线	20
2.4.1 连接节点	20
2.4.2 连线样式	21
2.5 Graph 指令	22

II 语法

一、Node	23
1.1 节点基础	23
1.1.1 Node 命令的语法	23
1.1.2 盒模型	26
1.1.3 边框形状	27
1.2 节点样式	28
1.2.1 分割节点	28
1.2.2 节点文字	28
1.2.3 节点文本	29
1.2.4 节点变换	30
1.2.5 节点复用	31
1.3 节点布局	31
1.3.1 定位	31
1.3.2 高阶布局	32
1.3.3 节点集	34
1.4 节点交互	34
1.4.1 曲线上的节点	34
1.4.2 节点之间交互	37
1.4.3 节点图像	37
1.5 节点拓展	38
1.5.1 节点标签	38
1.5.2 大头针	39
1.5.3 边缘	40

III 百科

一、参数百科	42
1.1 通用参数	42
1.1.1 常见通用参数意义	42
1.2 TikZ 特有参数	42
1.2.1 对齐: alignment option	42
1.2.2 锚点: anchor name	43
1.2.3 偏移: offset	43

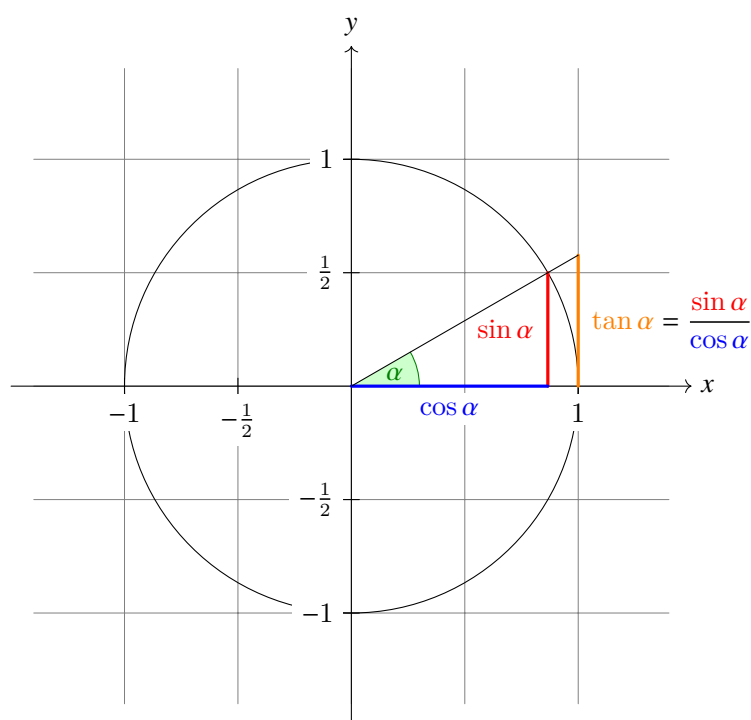
I 案例

次章节源于 TikZ 手册¹ 的学习

一、三角函数：A Picture for Karl's Students

1.1 效果预览

这个案例的最终效果如下



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

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

By the Theorem of Pythagoras ...

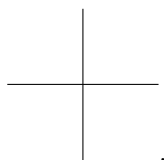
图 1.1.1

启用 tikz 环境：

```
\begin{tikzpicture}
.....
\end{tikzpicture}
```

画坐标轴：

¹TikZ Github 仓库：TikZGithub <https://github.com/pgf-tikz/pgf>



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

图 1.1.2 直线

两种 TikZ 画法:

1. 通过 TikZ 环境，如上所述
2. 行内 TikZ，如 _____

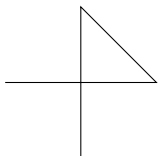
在不同的环境下 TikZ 有多种写法，这里不做说明

1.2 基本曲线

1.2.1 直线

直线通过 `\draw` 指令绘制:

```
\draw (x,y) -- (x,y) -- (x,y) ... ;
```



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

图 1.1.3 直线

1.2.2 曲线

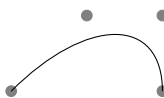
TikZ 的曲线原理在官方文档中没有详细解释，类似于贝塞尔曲线，我的理解如下:

- 和直线类似，曲线也由点组成，和分为起止点，过程点
- 起止点为曲线的起点和终点，必须经过
- 过程点比必须经过，曲线有趋近的趋势

通过 `\.. controls` 关键字控制曲线

```
Official: <start point> .. controls <1st control point> and <2nd control point> .. <end point>;
Simple: \draw (x,y) .. controls (x,y) and (x,y) .. (x,y);
```

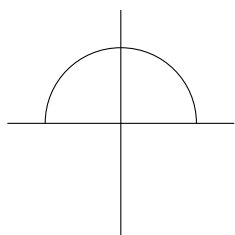
可以取消.. 这样会让第一个点使用两次



```
\begin{tikzpicture}
  \filldraw [gray] (0,0) circle [radius=2pt]
    (1,1) circle [radius=2pt]
    (2,1) circle [radius=2pt]
    (2,0) circle [radius=2pt];
  \draw (0,0) .. controls (1,1) and (2,1) .. (2,0);
\end{tikzpicture}
```

图 1.1.4 曲线

继续坐标系的例子，可以画出伪半圆，注意下面的例子中，(0,1) 点既作为终点又作为起点



```
\begin{tikzpicture}
\draw (-1.5,0) -- (1.5,0);
\draw (0,-1.5) -- (0,1.5);
\draw (-1,0) .. controls (-1,0.555) and (-0.555,1) .. (0,1)
.. controls (0.555,1) and (1,0.555) .. (1,0);
\end{tikzpicture}
```

图 1.1.5 带半圆曲线的坐标轴

1.2.3 椭圆

可以通过如下方式绘制椭圆

```
\draw <point> <category> [args];
```

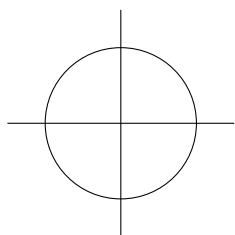
其中：<point>: 圆心 <category>: 椭圆类型 [args]: 对应的参数例如下面绘制的圆和椭圆



```
\draw (0,0) circle [radius=10pt];
\draw (0,0) ellipse [x radius=20pt, y radius=10pt];
```

图 1.1.6 椭圆

继续在坐标轴上加上圆



```
\begin{tikzpicture}
\draw (-1.5,0) -- (1.5,0);
\draw (0,-1.5) -- (0,1.5);
\draw (0,0) circle [radius = 1cm];
\end{tikzpicture}
```

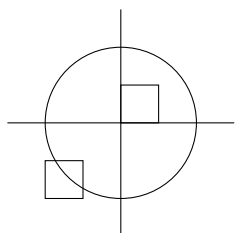
图 1.1.7 带单位圆的坐标轴

1.2.4 矩形

矩形的绘制方式和椭圆几乎一致，只需要将下面 category 写为 rectangle

```
\draw <point> <category> [args];
```

在上述坐标轴基础上继续绘制矩形



```
\begin{tikzpicture}
\draw (-1.5,0) -- (1.5,0);
\draw (0,-1.5) -- (0,1.5);
\draw (0,0) circle [radius=1cm];
\draw (0,0) rectangle (0.5,0.5);
\draw (-0.5,-0.5) rectangle (-1,-1);
\end{tikzpicture}
```

图 1.1.8 带矩形的单位圆

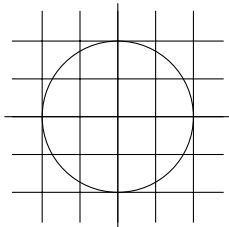
1.3 网格

1.3.1 网格的绘制

网格的绘制形式:

```
\draw[step=2pt] (0,0) grid (10pt,10pt);
```

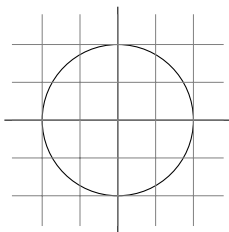
与曲线不同的是, 第一个坐标点表示网格一个角, 第二个坐标点则表示另一个不相邻的角, 由此, 我们可以在坐标轴基础上绘制网格背景了



```
\begin{tikzpicture}
  \draw (-1.5,0) -- (1.5,0);
  \draw (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius=1cm];
  \draw[step=.5cm] (-1.4,-1.4) grid (1.4,1.4);
\end{tikzpicture}
```

图 1.1.9 带网格的坐标轴

网格的颜色与单位圆相同, 不易区别, 这里使用更多的参数来控制



```
\begin{tikzpicture}
  \draw (-1.5,0) -- (1.5,0);
  \draw (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius=1cm];
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
\end{tikzpicture}
```

图 1.1.10 修改颜色的带网格的坐标轴

1.3.2 样式设置

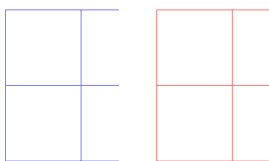
图像中的样式对于进场重复使用的样式, 我们可以设置一个统一样式, 如网格虚线, 我们定义为辅助线 help lines。

```
help lines/. style={color=blue!50,very thin}
```

全局样式除了在图像中定义样式供单个图像使用, 还可以定义全局样式。全局样式允许嵌套

```
\tikzset{help lines/.style=very thin}
\tikzset{Pionpill grid/.style={help lines,color = blue!50}}
```

样式同时可以设置参数



```
\begin{tikzpicture}
  [Karl's grid/.style = {help lines,color=#1!50},
  Karl's grid/.default=blue]
  \draw[Karl's grid] (0,0) grid (1.5,2);
  \draw[Karl's grid=red] (2,0) grid (3.5,2);
\end{tikzpicture}
```

图 1.1.11 带参数的样式

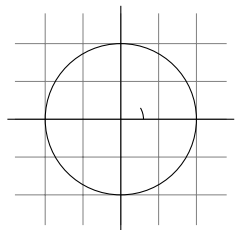
网格除了以上重要参数, 还有虚线参数可以设置 dash pattern

1.4 弧度

绘制弧度的形式：

```
\draw (x,y) arc [args]
```

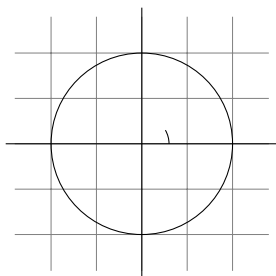
和网格类使，第一个坐标表示弧线起点，arc 为关键词，之后为样式参数，默认为逆时针旋转



```
\begin{tikzpicture}
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
  \draw(-1.5,0) -- (1.5,0);
  \draw(0,-1.5) -- (0,1.5);
  \draw(0,0) circle [radius=1cm];
  \draw(3mm,0mm) arc [start angle=0,end angle=30,radius=3mm];
\end{tikzpicture}
```

图 1.1.12 带弧度的单位圆

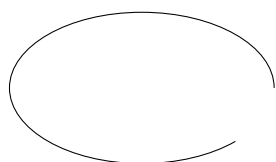
如果对图像大小不满意，可以使用 tikzpicture 的 scale 参数进行放大



```
\begin{tikzpicture}[scale=1.2]
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
  \draw(-1.5,0) -- (1.5,0);
  \draw(0,-1.5) -- (0,1.5);
  \draw(0,0) circle [radius=1cm];
  \draw(3mm,0mm) arc [start angle=0, end angle=30, radius=3
mm];
\end{tikzpicture}
```

图 1.1.13 放大后的图像

除了圆弧，还能绘制椭圆弧



```
\tikz \draw (0,0) arc [start angle=0, end angle=315, x radius
=1.75cm, y radius=1cm];
```

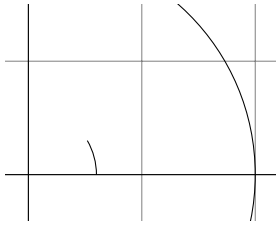
图 1.1.14 椭圆弧

1.5 截图

截图的用法与网格十分相识，可以指定截图方式，以及截图的对角来划定区域

```
\clip (x,y) graph (x,y)
```

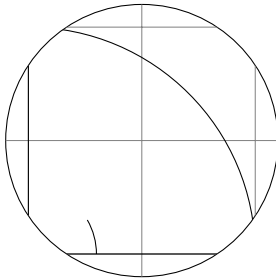
需要注意的是，clip 指令需要写在前面的位置，下面使用矩形截图：



```
\begin{tikzpicture}[scale = 3]
  \clip (-0.1,-0.2) rectangle (1.1,0.75);
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
  \draw (-1.5,0) -- (1.5,0);
  \draw (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius = 1cm];
  \draw (3mm,0mm) arc [start angle=0, end angle=30, radius=3
mm];
\end{tikzpicture}
```

图 1.1.15 矩形截图

下面结合 draw 绘制圆形截图，技术细节不在此讨论



```
\begin{tikzpicture}[scale=3]
  \clip[draw] (0.5,0.5) circle (.6cm);
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
  \draw (-1.5,0) -- (1.5,0);
  \draw (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius=1cm];
  \draw (3mm,0mm) arc [start angle=0, end angle=30, radius=3
mm];
\end{tikzpicture}
```

图 1.1.16 结合 draw 绘制圆形截图

1.6 抛物线与三角函数

1.6.1 抛物线

抛物线同样使用 draw 命令绘制



```
\tikz \draw (0,0) parabola (1,1)
```

图 1.1.17 抛物线

其他形式的抛物线



```
\tikz \draw[x=1pt,y=1pt] (0,0) parabola bend (4,16) (6,12)
```

图 1.1.18 其他形式抛物线

1.6.2 三角函数

三角函数同样使用 draw 指令绘制

```
\draw[x=,y=] (x,y) sin (x,y)
```

其中，可选项 $[x=,y=]$ 用于控制图像比例，第一个点为起点，随后指定三角函数类型，最后一个点为终点

⌒

```
\tikz \draw[x=1ex,y=1ex] (0,0) sin (1.57,1);
```

图 1.1.19 部分三角函数图像



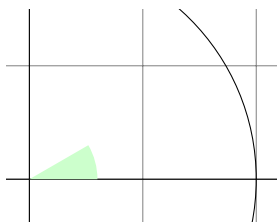
```
\tikz \draw [x=1.57ex,y=1ex] (0,0) sin(1,1) cos(2,0) sin(3,-1)
cos(4,0) (0,1) cos(1,0) sin(2,-1) cos(3,0) sin(4,1);
```

图 1.1.20 三角函数图像

1.7 填充与边线

1.7.1 基本填充与边线

使用 fill 代替 draw 关键字可绘制填充图形



```
\begin{tikzpicture}[scale = 3]
\clip (-0.1,-0.2) rectangle (1.1,0.75);
\draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
\draw (-1.5,0) -- (1.5,0);
\draw (0,-1.5) -- (0,1.5);
\draw (0,0) circle [radius=1cm];
\fill[green!20!white] (0,0) -- (3mm,0mm)
arc [start angle=0, end angle=30, radius=3mm] -- (0,0);
\end{tikzpicture}
```

图 1.1.21 填充图形

边框线: useasboundingbox

```
\useasboundingbox (low,high)
```

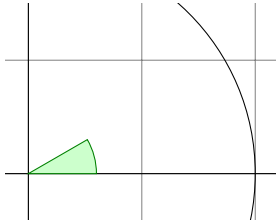
low 和 high 分别代表原边界向内，向外扩展，可使用 cycle 来指明曲线闭合



```
\begin{tikzpicture}[line width=5pt]
\draw (0,0) -- (1,0) -- (1,1) -- (0,0);
\draw (2,0) -- (3,0) -- (3,1) -- cycle;
\useasboundingbox (0,1.5); % make bounding box higher
\end{tikzpicture}
```

图 1.1.22 边框粗细与闭合

可以使用 filldraw 绘制有边界线和填充的图形



```
\begin{tikzpicture}[scale = 3]
  \clip (-0.1,-0.2) rectangle (1.1,0.75);
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
  \draw (-1.5,0) -- (1.5,0);
  \draw (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius=1cm];
  \filldraw[fill=green!20!white, draw=green!50!black] (0,0)
    -- (3mm,0mm) arc [start angle=0, end angle=30, radius
    =3mm] -- cycle;
\end{tikzpicture}
```

图 1.1.23 filldraw 效果

1.7.2 渐变

与填充类使，渐变也有两种画法 shade 与 shadefrom



```
\tikz \shade (0,0) rectangle (2,1) (3,0.5) circle (.5cm)
```

图 1.1.24 shade 效果

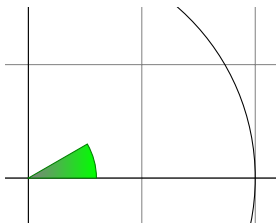
更详细的渐变设置



```
\begin{tikzpicture}[scale = 1]
  \shade [top color=yellow,bottom color=black] (0,0) rectangle + (2,1);
  \shade[left color=yellow,right color=black] (3,0) rectangle +(2,1);
  \shadedraw[inner color=yellow,outer color=black,draw=yellow] (6,0) rectangle +(2,1);
  \shade[ball color=green] (9,.5) circle (.5cm);
\end{tikzpicture}
```

图 1.1.25 几种渐变样式

尝试绘制新的圆弧



```
\begin{tikzpicture}[scale = 3]
  \clip (-0.1,-0.2) rectangle (1.1,0.75);
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
  \draw (-1.5,0) -- (1.5,0);
  \draw (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius=1cm];
  \shadedraw[left color=gray,right color=green, draw=green
    !50!black] (0,0) -- (3mm,0mm) arc [start angle=0, end
    angle=30, radius=3mm] -- cycle;
\end{tikzpicture}
```

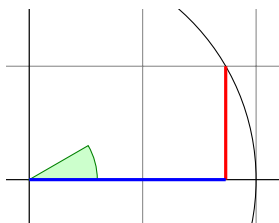
图 1.1.26 带渐变的圆弧

1.8 指定坐标

TikZ 指定坐标需要依靠数学计算，主要有两种方式

1. 平面直角坐标系：例如：(10pt,2cm) 代表 x 方向偏移 10pt, y 方向偏移 2cm
2. 极坐标系：例如：(30:1cm) 代表，逆时针选择 30°, 长度为 1cm

在确定坐标之后便可以依照新的点绘制图像，例如 +(0cm,1cm) 代表在坐标上方绘制 1cm 长线，++(2cm,0cm) 则再次绘制线段



```
\begin{tikzpicture}[scale = 3]
  \clip (-0.1,-0.2) rectangle (1.1,0.75);
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
  \draw (-1.5,0) -- (1.5,0);
  \draw (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius=1cm];
  \filldraw[fill=green!20,draw=green!50!black] (0,0) -- (3mm,0mm)
    arc [start angle=0, end angle=30, radius=3mm] -- cycle;
  \draw[red,very thick] (30:1cm) -- +(0,-0.5);
  \draw[blue,very thick] (30:1cm) ++(0,-0.5) -- (0,0);
\end{tikzpicture}
```

图 1.1.27 指定坐标绘制线段

从上述的例子可以发现，- 承担了指定线段类型的责任，若没有 - 则不会绘制线段，可以将这种用法用在定位点上，下面用几个例子详细说明 + 与 ++ 的作用



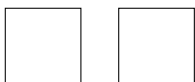
```
\begin{tikzpicture}[scale = 1]
  \def\rectanglepath{-- ++(1cm,0cm) -- ++(0cm,1cm) -- ++(-1cm,0cm) -- cycle}
  \draw (0,0) \rectanglepath;
  \draw (1.5,0) \rectanglepath;
\end{tikzpicture}
```

图 1.1.28 ++ 绘制矩形



```
\begin{tikzpicture}[scale = 1]
  \def\rectanglepath{-- +(1cm,0cm) -- +(0cm,1cm) -- +(-1cm,0cm) -- cycle}
  \draw (0,0) \rectanglepath;
  \draw (1.5,0) \rectanglepath;
\end{tikzpicture}
```

图 1.1.29 + 绘制矩形



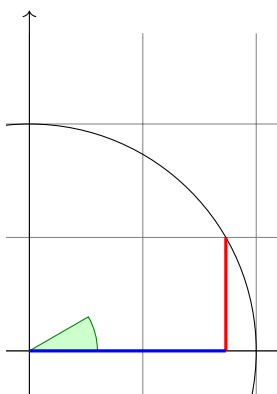
```
\tikz \draw (0,0) rectangle +(1,1) (1.5,0) rectangle +(1,1);
```

图 1.1.30 更快速的矩形绘制方式

1.9 线段与箭头

1.9.1 箭头样式

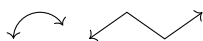
使用 -> 代替 - 绘制有箭头的坐标轴



```
\begin{tikzpicture}[scale = 3]
  \clip (-0.1,-0.2) rectangle (1.1,1.51);
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
  \draw[->] (-1.5,0) -- (1.5,0);
  \draw[->] (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius=1cm];
  \filldraw[fill=green!20,draw=green!50!black] (0,0) -- (3mm
    ,0mm) arc [start angle=0, end angle=30, radius=3mm] --
    cycle;
  \draw[red,very thick] (30:1cm) -- +(0,-0.5);
  \draw[blue,very thick] (30:1cm) ++(0,-0.5) -- (0,0);
\end{tikzpicture}
```

图 1.1.31 带箭头的坐标轴

可以在 draw 可选参数中使用箭头来指明箭头



```
\begin{tikzpicture}[scale = 1]
  \draw [<->] (0,0) arc [start angle = 180,end angle = 30,
    radius=10pt];
  \draw [<->] (1,0) -- (1.5cm,10pt) -- (2cm,0pt) -- (2.5cm,10
    pt);
\end{tikzpicture}
```

图 1.1.32 箭头示例

使用不一样的箭头样式



```
\usetikzlibrary {arrows.meta}
\begin{tikzpicture}[scale = 1,>=Stealth]
  \draw [->] (0,0) arc [start angle=180, end angle=30,radius
    =10pt];
  \draw [<<- ,very thick] (1,0) -- (1.5cm,10pt) -- (2cm,0pt)
    -- (2.5cm,10py);
\end{tikzpicture}
```

图 1.1.33 箭头样式

1.9.2 范围 scope

对于经常出现的同类样式，可以用 scope 设为统一样式



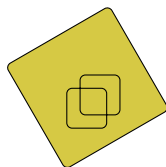
```
\begin{tikzpicture}[scale = 1,ultra thick]
  \draw (0,0) -- (0,1);
  \begin{scope}[thin]
    \draw (1,0) -- (1,1);
    \draw (2,0) -- (2,1);
  \end{scope}
  \draw (3,0) -- (3,1);
\end{tikzpicture}
```

图 1.1.34 scope 的作用

scope 还有一个重要作用，在 scope 内的操作只会影响内部范围

1.10 位移

可以使用 shift 关键字调整位置



```
\begin{tikzpicture}[scale = 1,rounded corners=2pt,x=15pt,y=15pt]
  \filldraw[fill=yellow!80!black] (0,0) rectangle (1,1)
    [xshift=5pt,yshift=5pt] (0,0) rectangle (1,1)
    [rotate=30] (-1,-1) rectangle (2,2);
\end{tikzpicture}
```

图 1.1.35 位移示范

1.11 循环

1.11.1 数字循环

使用 foreach 指令，进行数字循环

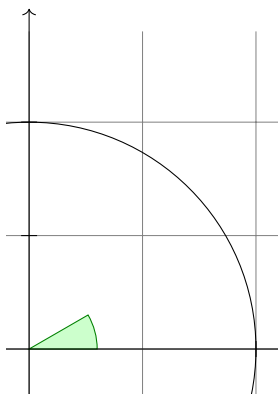
```
\foreach <variable> in <values> <commands>
```

$x = 1, x = 2, x = 3,$

```
\begin{tikzpicture}[scale = 1]
  \foreach \x in {1,2,3} {\x = \x$;}
\end{tikzpicture}
```

图 1.1.36 foreach 数字循环

同样，我们可以将其应用到绘图中



```
\begin{tikzpicture}[scale = 3]
  \clip (-0.1,-0.2) rectangle (1.1,1.51);
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
  \filldraw[fill=green!20,draw=green!50!black] (0,0) -- (3mm,0mm) arc [
    start angle=0, end angle=30, radius=3mm] -- cycle;
  \draw[->] (-1.5,0) -- (1.5,0);
  \draw[->] (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius=1cm];
  \foreach \x in {-1cm,-0.5cm,1cm} \draw (\x,-1pt) -- (\x,1pt);
  \foreach \y in {-1cm,-0.5cm,0.5cm,1cm} \draw (-1pt,\y) -- (1pt,\y);
\end{tikzpicture}
```

图 1.1.37 循环绘制刻度线

再进一步，我们可以设计两个变量，绘制二维矩阵

1,5	2,5	3,5	4,5	5,5	7,5	8,5	9,5	10,5	11,5	12,5
1,4	2,4	3,4	4,4	5,4	7,4	8,4	9,4	10,4	11,4	12,4
1,3	2,3	3,3	4,3	5,3	7,3	8,3	9,3	10,3	11,3	12,3
1,2	2,2	3,2	4,2	5,2	7,2	8,2	9,2	10,2	11,2	12,2
1,1	2,1	3,1	4,1	5,1	7,1	8,1	9,1	10,1	11,1	12,1

```

\begin{tikzpicture}[scale = 1]
  \foreach \x in {1,2,...,5,7,8,...,12}
    \foreach \y in {1,...,5} {
      \draw (\x,\y) +(-.5,-.5) rectangle ++(.5,.5);
      \draw (\x,\y) node{\x,\y};
    }
\end{tikzpicture}

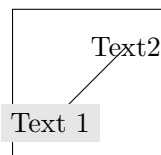
```

图 1.1.38 二维矩阵

1.12 增加文字

1.12.1 node 关键字

TikZ 可以通过 node 关键字指定节点并绘制区域，以达到添加文字的效果



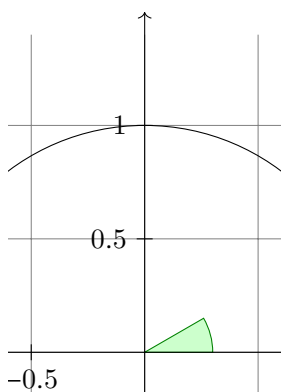
```

\begin{tikzpicture}[scale = 1]
  \draw (0,0) rectangle (2,2);
  \draw (0.5,0.5) node [fill=grey!20] {Text 1} -- (1.5,1.5)
    node {Text 2}
\end{tikzpicture}

```

图 1.1.39 node 关键字的作用

默认将以 node 为几何中心构建方形区域，但也可自定义文字位置



```

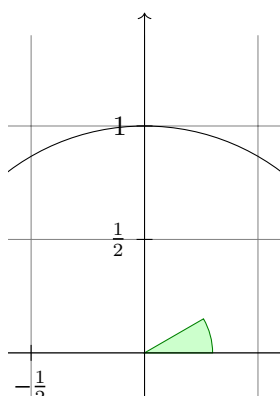
\begin{tikzpicture}[scale = 3]
  \clip (-0.6,-0.2) rectangle (0.6,1.51);
  \draw[step=.5cm,help lines] (-1.4,-1.4) grid (1.4,1.4);
  \filldraw[fill=green!20,draw=green!50!black] (0,0) -- (3mm,0mm)
    arc [start angle=0, end angle=30, radius=3mm] -- cycle;
  \draw[>-] (-1.5,0) -- (1.5,0); \draw[>-] (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius=1cm];
  \foreach \x in {-1,-0.5,1}
    \draw (\x cm,1pt) -- (\x cm,-1pt) node[anchor=north] {$\x$};
  \foreach \y in {-1,-0.5,0.5,1}
    \draw (1pt,\y cm) -- (-1pt,\y cm) node[anchor=west] {$\y$};
\end{tikzpicture}

```

图 1.1.40 绘制坐标系制度

如果需要使用分数形式的文本，可以使用如下形式

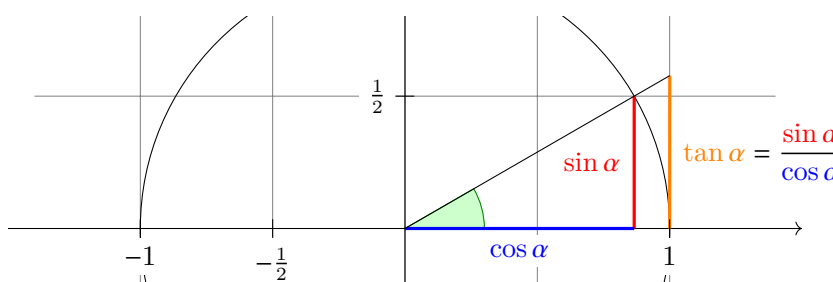
```
\x / \xtext
```

```
\begin{tikzpicture}[scale = 3]
  \clip (-0.6,-0.2) rectangle (0.6,1.51);
  \draw[step=.5cm,help lines] (-1.4,-1.4) grid (1.4,1.4);
  \filldraw[fill=green!20,draw=green!50!black] (0,0) -- (3mm,0mm) arc [
    start angle=0, end angle=30, radius=3mm] -- cycle;
  \draw[->] (-1.5,0) -- (1.5,0); \draw[->] (0,-1.5) -- (0,1.5);
  \draw (0,0) circle [radius=1cm];
  \foreach \x/\xtext in {-1, -0.5/\frac{1}{2}, 1}
    \draw (\x cm,1pt) -- (\x cm,-1pt) node[anchor=north] {$\xtext$};
  \foreach \y/\ytext in {-1, -0.5/\frac{1}{2}, 0.5/\frac{1}{2}, 1}
    \draw (1pt,\y cm) -- (-1pt,\y cm) node[anchor=east] {$\ytext$};
\end{tikzpicture}
```

图 1.1.41 绘制坐标系制度 (分数形式)

接着之前的直角坐标系，我们可以标出函数 (以下为官方代码，出现了部分前面未说明的内容)

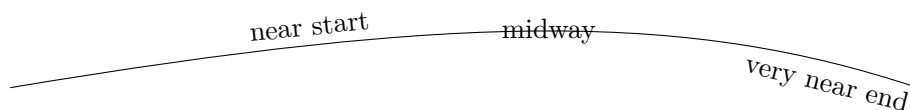


```
\begin{tikzpicture}[scale=3.5]
  \clip (-2,-0.22) rectangle (2,0.8);
  \draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
  \filldraw[fill=green!20,draw=green!50!black] (0,0) -- (3mm,0mm) arc [start angle=0, end angle=30,
    radius=3mm] -- cycle;
  \draw[->] (-1.5,0) -- (1.5,0) coordinate (x axis);
  \draw[->] (0,-1.5) -- (0,1.5) coordinate (y axis);
  \draw (0,0) circle [radius=1cm];
  \draw[very thick,red] (30:1cm) -- node[left=1pt,fill=white] {$\sin \alpha$} (30:1cm |- x axis);
  \draw[very thick,blue] (30:1cm |- x axis) -- node[below=2pt,fill=white] {$\cos \alpha$} (0,0);
  \path [name path=upward line] (1,0) -- (1,1);
  \path [name path=sloped line] (0,0) -- (30:1.5cm);
  \draw [name intersections={of=upward line and sloped line, by=t}] [very thick,orange] (1,0) -- node [
    right=1pt,fill=white] {$\displaystyle \tan \alpha \color{black}= \frac{\color{red}\sin \alpha}{\color{blue}\cos \alpha}$} (t);
  \draw (0,0) -- (t);
  \foreach \x/\xtext in {-1, -0.5/\frac{1}{2}, 1}
    \draw (\x cm,1pt) -- (\x cm,-1pt) node[anchor=north,fill=white] {$\xtext$};
  \foreach \y/\ytext in {-1, -0.5/\frac{1}{2}, 0.5/\frac{1}{2}, 1}
    \draw (1pt,\y cm) -- (-1pt,\y cm) node[anchor=east,fill=white] {$\ytext$};
\end{tikzpicture}
```

图 1.1.42 带文字的单位圆

1.12.2 线段文字

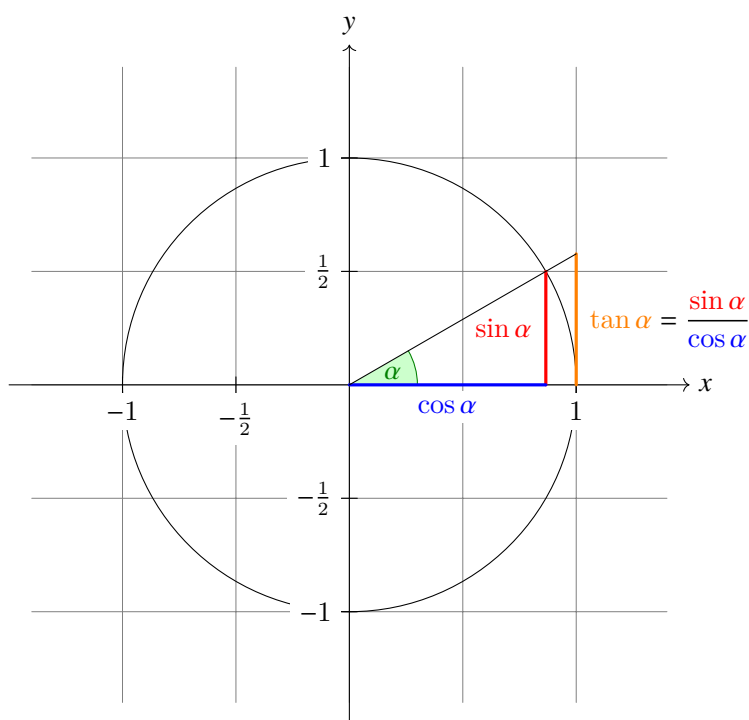
可以通过 `sloped` 关键字指定在线段的某一区域添加文字



```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) .. controls (6,1) and (9,1) ..
    node [near start,sloped,above] {near start}
    node {midway}
    node [very near end,sloped,below] {very near end}(12,0);
\end{tikzpicture}
```

图 1.1.43 sloped 控制段落文字位置

1.13 最终结果



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

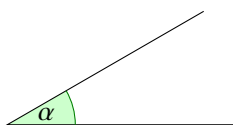
$$\sin \alpha = 1/2.$$

By the Theorem of Pythagoras ...

图 1.1.44

1.14 其他技巧

可以使用 `coordinate` 指定点位置并赋予别名，并通过 `pic` 来进行快速绘图，`pic` 这里只做演示，详细内容后续章节会讲解



```
\usetikzlibrary {angles,quotes}
\begin{tikzpicture}[scale = 3]
  \coordinate (A) at (1,0);
  \coordinate (B) at (0,0);
  \coordinate (C) at (30:1cm);
  \draw (A) -- (B) -- (C)
    pic [draw=green!50!black, fill=green!20, angle radius=9mm,
        "$\alpha$"] {angle = A--B--C};
\end{tikzpicture}
```

图 1.1.45 使用 pic 快速绘制

二、流程图：Diagrams as Simple Graphs

2.1 效果预览

这个案例的最终效果如下

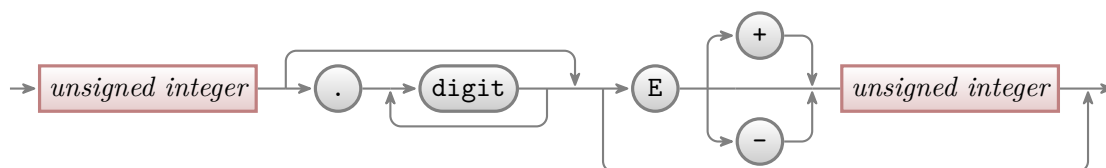


图 1.2.1 流程图效果预览

2.2 节点设置

2.2.1 节点样式

可以使用关键词 `/.style` 设置一个同一的样式

```
\begin{tikzpicture}[
  nonterminal/.style={
    rectangle,
    minimum size=6mm,
    very thick,
    draw=red!50!black!50,
    top color=white,
    bottom color=red!50!black!20,
    font=\itshape
  }
]
\node [nonterminal] {unsigned integer};
\end{tikzpicture}
```

图 1.2.2 基本节点样式

使用 `rounded corner` 进行圆角设置:



```
\begin{tikzpicture}[
  node distance = 5mm, terminal/.style = {
    rectangle,minimum size=6mm,rounded corners = 3mm,
    very thick, draw = black!50,
    top color = white, bottom color = black!20,
    font = \ttfamily
  }
]
\node (dot) [terminal] {.};
\node (digit) [terminal,right=of dot] {digit};
\node (E) [terminal,right=of digit] {E};
\end{tikzpicture}
```

图 1.2.3 节点样式：圆角

使用 `shapes.misc` 包的圆角设置:`rounded rectangle`



```
\usetikzlibrary {positioning}
\begin{tikzpicture}[node distance=5mm,
  terminal/.style={
    rounded rectangle,
    minimum size=6mm,
    very thick,draw=black!50,
    top color=white,bottom color=black!20,
    font=\ttfamily
  }]
\node (dot) [terminal] {.};
\node (digit) [terminal,right=of dot] {digit};
\node (E) [terminal,right=of digit] {E};
\end{tikzpicture}
```

图 1.2.4 rounded 圆角节点

在上述的几个例子中，我们发现. 与其他字符并没有对齐，. 看起来更像 •。这是由于每个节点内容拥不同的高度与字母深度。可以使用 `text height`, `text depth` 来进行调整

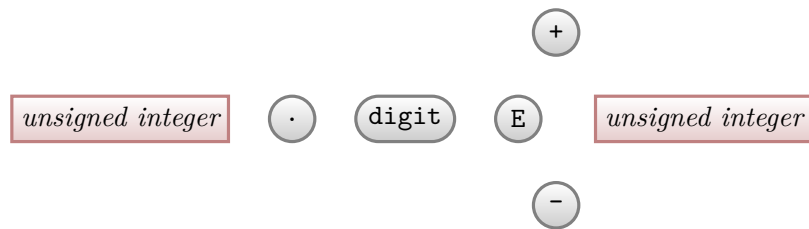


```
\begin{tikzpicture}[
  node distance=5mm, text height=1.5ex, text depth = .25ex,
  terminal/.style={
    rounded rectangle,
    minimum size=6mm,
    very thick,draw=black!50,
    top color=white,bottom color=black!20,
    font=\ttfamily
  }]
\node (dot) [terminal] {.};
\node (digit) [terminal,right=of dot] {digit};
\node (E) [terminal,right=of digit] {E};
\end{tikzpicture}
```

图 1.2.5 文字对齐

2.2.2 节点位置

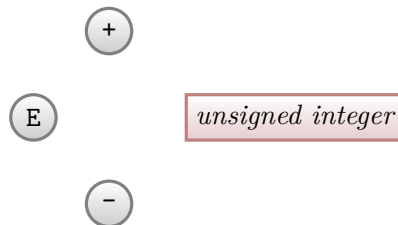
使用位置选项 `position options` 设置节点相对位置



```
\begin{tikzpicture}[node distance=5mm and 5mm]
  \node (ui1) [nonterminal] {unsigned integer};
  \node (dot) [terminal,right=of ui1] {.};
  \node (digit) [terminal,right=of dot] {digit};
  \node (E) [terminal,right=of digit] {E};
  \node (plus) [terminal,above right=of E] {+};
  \node (minus) [terminal,below right=of E] {-};
  \node (ui2) [nonterminal,below right=of plus] {unsigned integer};
\end{tikzpicture}
```

图 1.2.6 节点位置

精确控制偏移距离的方法之一：使用 xshift yshift



```
\begin{tikzpicture}[scale = 1,node distance = 5mm and 5mm]
  \node (E) [terminal] {E};
  \node (plus) [terminal,above right=of E,xshift=5mm] {+};
  \node (minus) [terminal,below right=of E,xshift=5mm] {-};
  \node (ui2) [nonterminal,below right=of plus,xshift=5mm] {
    unsigned integer};
\end{tikzpicture}
```

图 1.2.7 节点位置精确控制

2.2.3 节点连线

tikz 中的节点连接，尤其是非直线连接是一件非常复杂的事情。



```
\begin{tikzpicture}[node distance=5mm and 5mm]
  \node (dot) [terminal] {.};
  \node (digit) [terminal,right=of dot] {digit};
  \node (E) [terminal,right=of digit] {E};
  \path (dot) edge[->] (digit) % simple edges
        (digit) edge[->] (E);
  \draw [->]
    % start right of digit.east, that is, at the point that
    % is the linear combination of digit.east and the
    % vector (2mm,0pt). We use the ($ ... $) notation for
    % computing linear combinations
    ($ (digit.east) + (2mm,0) $)
    % Now go down
    -- ++(0,-.5)
    % And back to the left of digit.west
    -| ($ (digit.west) - (2mm,0) $);
\end{tikzpicture}
```

图 1.2.8 节点连线

在此基础上，tikz 引入了 edge 参数来简化折线。

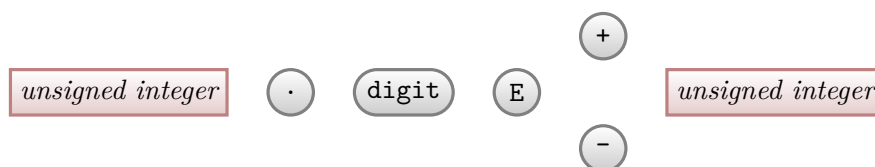


```
\begin{tikzpicture}[
  node distance=5mm and 5mm,skip loop/.style={to path={--
    ++(0,-.5) -| (\tikztotarget)}}]
  \node (dot) [terminal] {.};
  \node (digit) [terminal,right=of dot] {digit};
  \node (E) [terminal,right=of digit] {E};
  \path (dot) edge[->] (digit) % simple edges
    (digit) edge[->] (E)
    ($ (digit.east) + (2mm,0) $)
    edge[->,skip loop] ($ (digit.west) - (2mm,0) $);
\end{tikzpicture}
```

图 1.2.9 简化的节点连线

2.3 矩阵布局

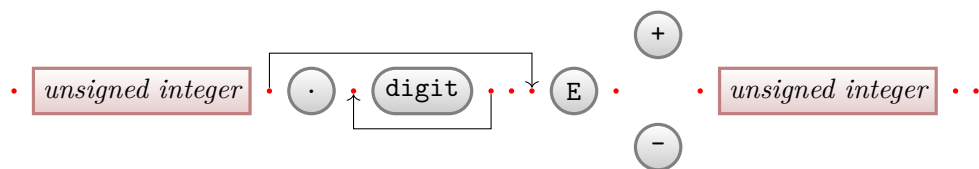
矩阵式布局与 latex 自身的数学公式布局十分相似



```
\begin{tikzpicture}
  \matrix[row sep=1mm,column sep=5mm] {
    % First row:
    & & & \node [terminal] {+}; & \\
    % Second row:
    \node [nonterminal] {unsigned integer}; &
    \node [terminal] {.}; &
    \node [terminal] {digit}; &
    \node [terminal] {E}; &
    \node [nonterminal] {unsigned integer}; & \\
    % Third row:
    & & & \node [terminal] {-}; & \\
  };
\end{tikzpicture}
```

图 1.2.10 矩阵布局

使用了矩阵布局后，连线将变得非常简单



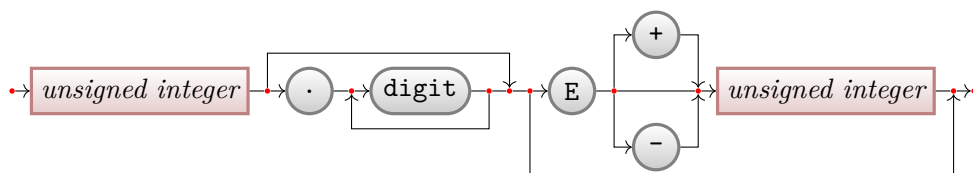
```
\begin{tikzpicture}
  [point/.style={circle,inner sep=0pt,minimum size=2pt,fill=red},skip loop/.style={to path
    ={{-- ++(0,#1) -| (\tikztotarget)}}}]
  \matrix[row sep=1mm,column sep=2mm] {
    % First row:
    & & & & & & & & \node (plus) [terminal] {+};\\
    % Second row:
    \node (p1) [point] {}; & \node (ui1) [nonterminal] {unsigned integer}; & \\
    \node (p2) [point] {}; & \node (dot) [terminal] {.}; & \\
    \node (p3) [point] {}; & \node (digit) [terminal] {digit}; & \\
    \node (p4) [point] {}; & \node (p5) [point] {}; & \\
    \node (p6) [point] {}; & \node (e) [terminal] {E}; & \\
    \node (p7) [point] {}; & & \\
    \node (p8) [point] {}; & \node (ui2) [nonterminal] {unsigned integer}; & \\
    \node (p9) [point] {}; & \node (p10) [point] {}; & \\
    % Third row:
    & & & & & & & & \node (minus)[terminal] {-};\\
  };
  \path (p4) edge [->,skip loop=-5mm] (p3)
        (p2) edge [->,skip loop=5mm] (p6);
\end{tikzpicture}
```

图 1.2.11 矩阵布局与连线

2.4 连线

2.4.1 连接节点

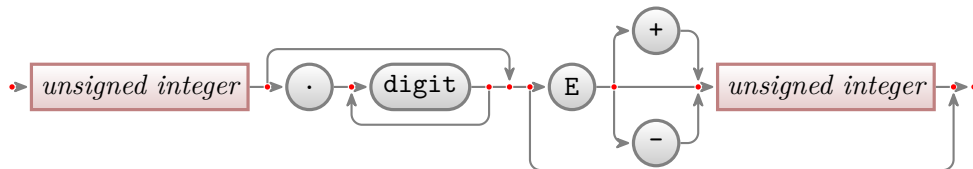
使用 `graph` 命令 (`path graph` 的简写) 绘制连线



```
\begin{tikzpicture}
  [point/.style={circle,inner sep=0pt,minimum size=2pt,fill=red},
  skip loop/.style={to path={-- ++(0,#1) -| (\tikztotarget)}},
  hv path/.style={to path={-| (\tikztotarget)}},
  vh path/.style={to path={|- (\tikztotarget)}}]
  \matrix[row sep=1mm,column sep=2mm] {
    .....
  };
  \graph {
    (p1) -> (ui1) -- (p2) -> (dot) -- (p3) -> (digit) -- (p4)
      -- (p5) -- (p6) -> (e) -- (p7) -- (p8) -> (ui2) -- (p9) -> (p10);
    (p4) -> [skip loop=-5mm] (p3);
    (p2) -> [skip loop=5mm] (p5);
    (p6) -> [skip loop=-11mm] (p9);
    (p7) -> [vh path] (plus) -> [hv path] (p8);
    (p7) -> [vh path] (minus) -> [hv path] (p8);
  };
\end{tikzpicture}
```

图 1.2.12 graph 连线

2.4.2 连线样式



```
\begin{tikzpicture}
  [>={Stealth[round]},thick,black!50,text=black,
  every new ->/.style={shorten >=1pt},
  graphs/every graph/.style={edges=rounded corners},
  point/.style={circle,inner sep=0pt,minimum size=2pt,fill=red},
  skip loop/.style={to path={-- ++(0,#1) -| (\tikztotarget)}},
  hv path/.style={to path={-| (\tikztotarget)}},
  vh path/.style={to path={|- (\tikztotarget)}}]
  \matrix[row sep=1mm,column sep=2mm] {
    .....
  };
  \graph [use existing nodes] {
    p1 -> ui1 -- p2 -> dot -- p3 -> digit -- p4 -- p5 -- p6 -> e -- p7 -- p8 -> ui2 -- p9
      -> p10;
    p4 ->[skip loop=-5mm] p3;
    p2 ->[skip loop=5mm] p5;
    p6 ->[skip loop=-11mm] p9;
    p7 ->[vh path] { plus, minus } -> [hv path] p8;
  };
\end{tikzpicture}
```

图 1.2.13 连线样式

2.5 Graph 指令

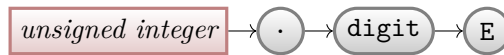
使用 graph 节点可以快速绘制简单的流程图

unsigned integer \rightarrow d \longrightarrow digit \longrightarrow E

```
\tikz \graph [grow right=2cm] {unsigned integer -> d -> digit -> E};
```

图 1.2.14 graph 指令绘图

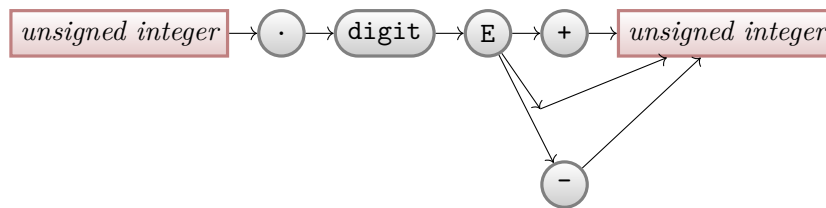
声明 grow right sep 指明让节点在前一节点右方生成



```
\tikz \graph [grow right sep] {  
  unsigned integer[nonterminal] -> "."[terminal] -> digit[terminal] -> E[terminal]  
};
```

图 1.2.15

创建分支



```
\tikz \graph [grow right sep] {  
  unsigned integer [nonterminal] ->  
  "." [terminal] ->  
  digit [terminal] ->  
  E [terminal] ->  
  {  
    "+" [terminal],  
    "" [coordinate],  
    "-" [terminal]  
  } ->  
  ui2/unsigned integer [nonterminal]  
};
```

图 1.2.16

II 语法

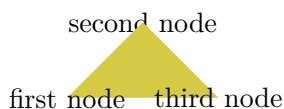
一、Node

1.1 节点基础

Node 是一个带有文字的简单图形 (矩形, 圆形, 点)。Node 不属于 path, 通常在 path 已经绘制好后, 或绘制之前产生。

Node 的最简单用法是在一些坐标点周围添加文字。与此同时, Node 也能添加图案以及复杂的颜色效果, 甚至一些 Node 取消了文字。

添加 Node 并没有专用的 L^AT_EX 语法, 通常用在 path 中用 node 指明。



```
\begin{tikzpicture}[scale = 1]
  \fill [fill=yellow!80!black]
    (0,0) node {first node}
    -- (1,1) node[behind path] {second node}
    -- (2,0) node {third node};
\end{tikzpicture}
```

图 2.1.1 Node 基本用法

1.1.1 Node 命令的语法

在 path 中添加 node 的完整语法如下:

```
\path ... node <foreach statements> [<options>] (<name>) at(<coordinate>) :(<animation attribute>)=<options> {<node contents>} ...;
```

另一种轻量化的写法

```
\node [<options>] (<name>) at(<coordinate>) :(<animation>);
```

主要用法

```
... node [options] {text} ...;
```

在 text 中添加 node 要显示的文字, [options] 指定样式, 下面整理常用 [options]。

- 文字与颜色

除了在 text 中指明颜色, 在 [options] 也可以使用 [node contents=<text>] 指定文字, 且在 [options] 中说明的颜色默认为文字颜色。

A B C D

```
\begin{tikzpicture}[scale = 1]
  \path (0,0) node [blue] {A}
        (1,0) node [red] {B}
        (2,0) node [green,node contents=C]
        (3,0) node [node contents=D];
\end{tikzpicture}
```

图 2.1.2 Node 的文字与颜色

- 指定位置

平面位置: [at=<coordinate>]: 指定 node 的位置, 当 node 在 path 中时, 无效。

图层位置: [behind path]: 指定 node 的图层位置, 效果见图2.1.1。默认样式为 [in front of path]

- 节点名

节点名用于后续绘图指定节点, TikZ 允许至多两个节点名。

节点名: [name=<name>]: 节点的名称

节点别名: [alias=<alias>]: 节点别名

- 节点形状

默认节点仅有文字, 不具有形状, 需要使用 [draw/fill] 命令指定对应形状。

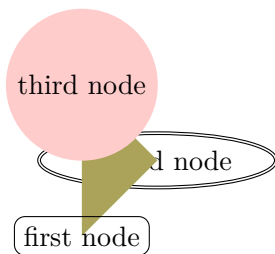
边框: [draw]: 显示节点边线;

底色: [fill=<color>]: 显示节点底色

边框形状: [shape = rectangle, circle, ellipse]: shape 可以省略, 默认为矩形 rectangle, 更多形状可查阅官方资料。

边框圆角: [rounded corners]

边线数量: [double]



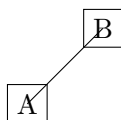
```
\begin{tikzpicture}[scale = 1]
  \fill[fill=yellow!60!black]
    (0,0) node [draw, rounded corners] {first node}
    -- (1,1) node [draw, double, behind path] {second node}
    -- (0,2) node [circle,fill=red!20] {third node};
\end{tikzpicture}
```

图 2.1.3 Node 形状

- 节点全局样式

可以通过在 tikzpicture 环境开始处添加说明, 给予全局节点样式。

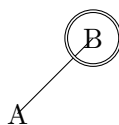
全部样式: [every node/.style = {}]



```
\begin{tikzpicture}[every node/.style={draw}]
  \draw (0,0) node {A} -- (1,1) node {B};
\end{tikzpicture}
```

图 2.1.4 Node 全部样式

指定样式: [every <shape> node/.style = {}]



```
\begin{tikzpicture}[every circle node/.style={draw,double}]
  \draw (0,0) node {A}
    -- (1,1) [circle] node {B};
\end{tikzpicture}
```

图 2.1.5 Node 指定样式

文字前后缀: [execute at begin/end node = {text}]

第一题:

```
\begin{tikzpicture}[execute at begin node = {第}, execute at
  end node = {题}]
  \node [execute at end node = {: }] {-};
\end{tikzpicture}
```

图 2.1.6

- 其他样式

填充: [fill = <color>]: 填充背景色

缩放: [scale = <dimension>]: 节点缩放

边框粗细: [linewidth = <dimension>]: 边框粗细

其他用法

- 节点动画

通过:<animation attribute>=<options>, 可以设定动画¹, 下面只做简单举例



```
\begin{tikzpicture}[scale = 1]
  \node :fill opacity={0s="1",2s="0",begin on=click}
    :rotate = {0s="0",2s="90",begin on=click}
    [fill = blue!20, draw = blue, ultra thick, circle
      ]
    {click me};
\end{tikzpicture}
```

图 2.1.7 Node 动画

- foreach

foreach 语句仅允许紧跟在 node 之后出现。

语法形式: foreach \x in {}

在集合中可以使用... 表示省略类似的内容。

1 2 3

```
% 以下两句效果相同
\draw (0,0) node foreach \x in {1,2,3} at (\x,0) {\x};
\tikz \draw (0,0) node at (1,0) {1} node at (2,0) {2} node at
      (3,0) {3};
```

图 2.1.8 Node 一次迭代

¹部分图形驱动并不支持动画, 比如我的也不支持

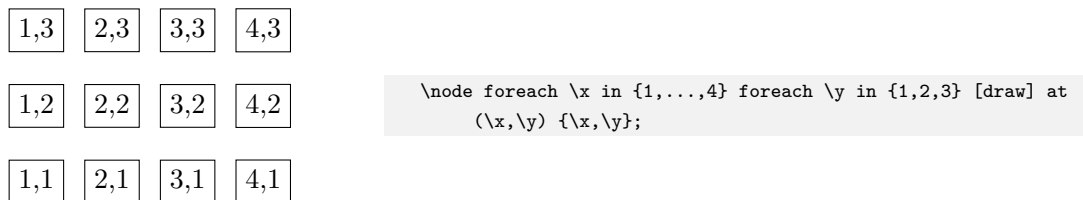


图 2.1.9 Node 两次迭代

- scope

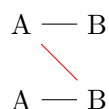
scope 用于限定范围，类似高级语言中的命名空间。

和 \LaTeX 中的环境十分相似，scope 需要 `\begin` 和 `\end` 来限定范围。

在 `\begin{scope}[name prefix = <text>]` 限定范围名。

使用时“name”+“node name”即可。

类似的，也可以使用 suffix。



```
\begin{tikzpicture}[scale = 1]
  \begin{scope}[name prefix = top-]
    \node (A) at (0,1) {A};
    \node (B) at (1,1) {B};
    \draw (A) -- (B);
  \end{scope}
  \begin{scope}[name prefix = bottom-]
    \node (A) at (0,0) {A};
    \node (B) at (1,0) {B};
    \draw (A) -- (B);
  \end{scope}
  \draw [red] (top-A) -- (bottom-B);
\end{tikzpicture}
```

图 2.1.10 Node:sep

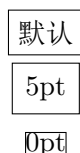
1.1.2 盒模型

盒模型²，这里指 Node 周围边距，底色等样式的控制。由于比一般的样式控制命令更多，而且重要，单独开一节做笔记。

- 边距 sep

(默认: 0.3333em)

总内边距: `[inner sep = <dimension>]`: 边框与内部文字的边距



```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) node [inner sep = 0pt,draw] {0pt}
    (0,2em) node [inner sep = 5pt,draw] {5pt}
    (0,4em) node [draw] {默认};
\end{tikzpicture}
```

图 2.1.11 Node:inner sep

左右/上下边距: `[inner xsep/ysep = <dimension>]`

外边距: `[outer sep = <dimension>]`, 外边距可能出现一些不准确的问题，可以在环境中加入 `[outer sep = auto]` 解决，与 inner sep 类使的，也可以指明 x/y 方向外边距。

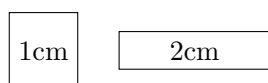
²盒模型概念来自 css，与这里极其类使，但 TikZ 官方并没有指定这一系列样式的名称

- 最小高度/宽度

最小高度/宽度用于限制节点与边线的最小距离

最小距离: [minimum size]: 最小高度与宽度

最小高度: [minimum height], 最小宽度: [minimum width]



```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) node [minimum height = 1cm,draw] {1cm}
        (2,0) node [minimum width = 2cm,draw] {2cm};
\end{tikzpicture}
```

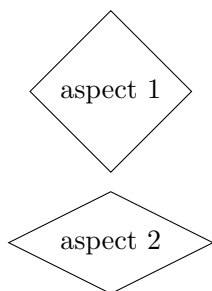
图 2.1.12 Node: 最小高度/宽度

1.1.3 边框形状

这里主要备注边框形状相关的内容。

- 横纵比

横纵比: [shape aspect=<aspect ratio>]: 外边框形状进行压缩。



```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) node [shape aspect=1,diamond,draw] {aspect
    1};
  \draw (0,-2) node [shape aspect=2,diamond,draw] {aspect
    2};
\end{tikzpicture}
```

图 2.1.13 Node: aspect

- 边框距

(默认: 1pt)

TikZ 的边框距有两种计算方式, 可以使用 [shape border uses incircle = <boolean>] 启动第二种边框距计算, 效果见下图:

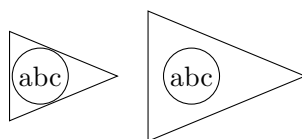
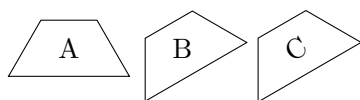


图 2.1.14 Node: 边距

- 旋转

文字旋转: [rotate = <angle>]: 文字和边框都将出现旋转。

边框旋转: [rotate = <angle>]: 仅边框旋转。



```
\begin{tikzpicture}[every node/.style={shape=trapezium, draw,
    shape border usincircle}]
  \node at (0,0) (A) {A};
  \node [shape border rotate=30] at (1.5,0) {B};
  \node [rotate=30] at (3,0) {C};
\end{tikzpicture}
```

图 2.1.15 Node: 旋转

- 边框粗细

粗细: [linewidth = <dimension>]



```
\begin{tikzpicture}[scale = 1]
  \node [line width=2,draw] at (0,0) {A};
\end{tikzpicture}
```

图 2.1.16 Node: 边框粗细

1.2 节点样式

1.2.1 分割节点

在 node 文字中使用 `\nodepart[<options>]{<part name>}` 可以对节点进行切割; 注意此时的 shape 形状后需加上 `split`, 否则无效。同时需要声明 `\use tikzlibrary shapes.multipart`



```
\begin{tikzpicture}[scale = 1]
  \node [circle split,draw,double] {$q_1$ \nodepart{lower}
    $00$};
\end{tikzpicture}
```

图 2.1.17 Node: 分割节点

对于批量修改节点样式, 可以使用类似 `every lower node part/.style = color` 的方法。



```
\begin{tikzpicture}[every lower node part/.style = red]
  \node [circle split,draw] {$q_1$ \nodepart{lower} $00$};
\end{tikzpicture}
```

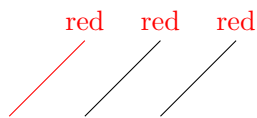
图 2.1.18 Node: 分割节点样式

1.2.2 节点文字

文字本身包含颜色, 字体, 大小等样式, 具体控制如下:

- 颜色

文字颜色: [color = <color>], 这里的 color 可以省略, 注意在节点中的颜色属性只影响节点文字, 这与 `\draw` 不同



```
\begin{tikzpicture}[scale = 1]
  \draw[red]      (0,0) -- +(1,1) node[above] {red};
  \draw[text = red] (1,0) -- +(1,1) node[above] {red};
  \draw          (2,0) -- +(1,1) node[above,red] {red};
\end{tikzpicture}
```

图 2.1.19 Node: 文字颜色

- 不透明度

不透明度: [opacity = <value>], 注意这里是不透明度, 1 表示完全不透明。

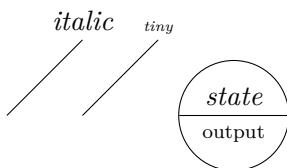


```
\begin{tikzpicture}[scale = 1]
  \draw[opacity = 1]      (0,0) -- +(1,1) node[above] {opacity};
  \draw                  (2,0) -- +(1,1) node[above,opacity = 0.1]
                        {opacity};
\end{tikzpicture}
```

图 2.1.20 Node: 文字透明度

- 文字字体

文字字体: [node font =], 这里的 node 可以省略。注意这里的 font 既可以指字体族, 也可以控制字体大小。



```
\begin{tikzpicture}[every text node part/.style={font=\itshape},
every lower node part/.style={font=\footnotesize}]
  \draw[node font=\itshape] (1,0) -- +(1,1) node[above] {
    italic};
  \draw[node font=\tiny] (2,0) -- +(1,1) node[above] {tiny};
  \node [circle split,draw] at (4,0) {state \nodepart{lower}
    output};
\end{tikzpicture}
```

图 2.1.21 Node: 文字字体

- 文字高度与深度

文字高度: [text height = <dimension>]

文字深度: [text depth = <dimension>]

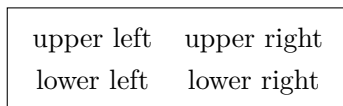
这两个属性并不常用, 一般情况下尽量用 inner sep 代替。

1.2.3 节点文本

文本格式包括文本框的长宽, 对齐方式等。

- 节点文本框

与正文中的文本类似, 节点中的文本也可以实现公式, 换行, 表格等功能。

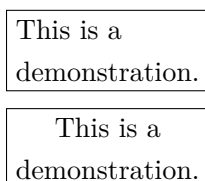


```
\begin{tikzpicture}[scale = 1]
  \node [draw] {
    \begin{tabular}{cc}
      upper left & upper right\\
      lower left & lower right
    \end{tabular}
  };
\end{tikzpicture}
```

图 2.1.22 Node: 节点文本框

- 文本对齐

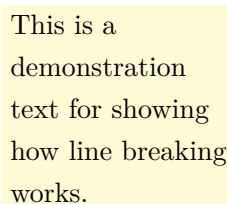
对齐: [align = <alignment option>], 设置对其方式。



```
\begin{tikzpicture}[scale = 1]
  \node[draw,align=left]at (0,0) {This is a\\demonstration.};
  \node[draw,align=center] at (0,-1.3) {This is a\\
    demonstration.};
\end{tikzpicture}
```

图 2.1.23 Node: 文本对齐

文本对齐会自动分割长单词，可以使用 align = flush <alignment option> 来取消长单词。



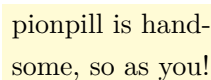
```
\begin{tikzpicture}[scale = 1]
  \node[fill=yellow!20,text width=3cm,align=flush left] {This
    is a demonstration text for showing how line breaking
    works.};
\end{tikzpicture}
```

图 2.1.24 Node: 分割长单词

alignment option 具体参数见属性百科:

- 文本宽度

文本宽: [text width = <dimension>]: 限定最大文本宽

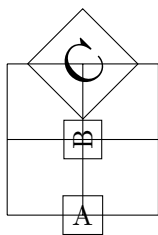


```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) node [text width=8em, fill = yellow!20] {
    pionpill is handsome, so as you!};
\end{tikzpicture}
```

图 2.1.25 Node: 文本宽度

1.2.4 节点变换

常用的节点的变换 (transform) 修饰有缩放与旋转。

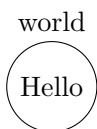


```
\begin{tikzpicture}[every node/.style = draw]
\draw (0,0) grid (2,2);
\draw (1,0) node {A};
\draw (1,1) node [rotate = 90] {B};
\draw (1,2) node [rotate = 45,scale = 2] {C};
\end{tikzpicture}
```

图 2.1.26 Node: 节点变换

1.2.5 节点复用

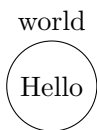
节点复用，即在使用过节点后，再使用节点。使用 `also` 关键字，即可启用之前定义过的节点。复用的节点无法添加文字。



```
\begin{tikzpicture}[scale = 1]
\node [circle,draw] (a) {Hello};
\node also [label=above:world] (a);
\end{tikzpicture}
```

图 2.1.27 Node:also

类似的，也可以使用 `[late options = {}]` 达到同样的效果。



```
\begin{tikzpicture}[scale = 1]
\node [draw,circle] (a) {Hello};
\path [late options = {name=a,label=above:world}];
\end{tikzpicture}
```

图 2.1.28 Node:late option

1.3 节点布局

1.3.1 定位

节点的位置往往由与之相关的坐标决定，默认以坐标为中心生成节点，同时也可在坐标周围生成节点。

- 节点锚点

锚点: `[anchor = <anchor name>]`: 锚点属性决定了节点将在坐标的哪个方向生成。
这里 `anchor` 的意思是坐标位于节点的方向。

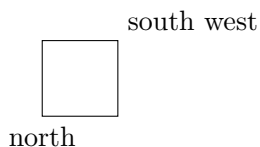


图 2.1.29 Node: 锚点

- 偏移

偏移: `[<offset>]`: 无需属性名，效果与 `anchor` 类似，但是能更精确地控制偏移量。

above
•
above
•

```
\begin{tikzpicture}[scale = 1]
  \fill (0,0) circle (2pt) node [above] {above};
  \fill (0,-1) circle (2pt) node [above=5pt] {above};
\end{tikzpicture}
```

图 2.1.30 Node: 偏移

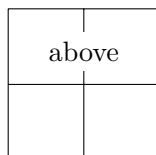
1.3.2 高阶布局

除了简单的定位，还可以使用 `positioning` 包来进行更为高阶的定位操作。在启动了该包后，原本的 `<dimension>` 参数将被提升为 `<specification>`。`<specification>` 参数通常由两部分组成：`<shifting part>` + `<of-part>`

`<shifting part>` 为主要控制参数，通常有三种形式：

- `<dimension>` 形式

这种形式在 `<dimension>` 参数的基础上，还可以使用数学式。

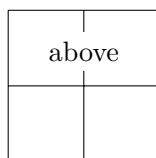


```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) grid (2,2);
  \node at (1,1) [above = 2pt+3pt,fill=white] {above};
\end{tikzpicture}
```

图 2.1.31 Node: 高阶布局-数学形式

- `<number>` 形式

这种形式可以不包含单位，其他和上面那个差别不大。

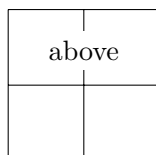


```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) grid (2,2);
  \node at (1,1) [above = .2,fill=white] {above};
  % south border of the node is now 2mm above (1,1)
\end{tikzpicture}
```

图 2.1.32 Node: 高阶布局-数字形式

- `and` 组合形式

可以通过 `and` 将多个偏移修饰组合，这里 `and` 的作用并不明显，下一节将详细说明。



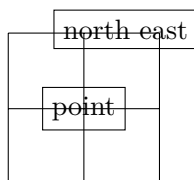
```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) grid (2,2);
  \node at (1,1) [above = .2 and 2mm,fill=white] {above};
  % south border of the node is also 2mm above (1,1)
\end{tikzpicture}
```

图 2.1.33 Node: 高阶布局-组合形式

`<of-part>` 可以进行相对定位，注意点如下：

- 以坐标为参照的定位

通过 `of` 关键字可以调用坐标点的方位进行相对定位。



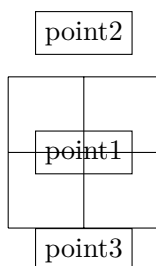
```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) grid (2,2);
  \node (point) [draw] at (1,1) {point};
  \node [above = 5mm of point.north east,draw] {north east};
\end{tikzpicture}
```

图 2.1.34 Node:of-part 坐标定位

定位逻辑：首先找到 point 的 north east 位置，然后向 above 方向移动了 5mm。
不止于坐标，一般别名都可以作为定位对象。

- 距离测算

默认状态下的偏移距离为边界之间的距离，而不是中心距。



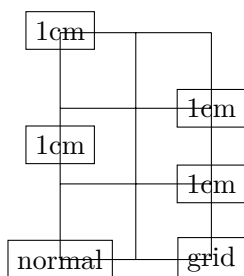
```
\begin{tikzpicture}[every node/.style = draw]
  \draw (0,0) grid (2,2);
  \node (point1) at (1,1) {point1};
  \node (point2) [above = 1cm of point1] {point2};
  \node (point3) [below = 1cm of point1.center] {point3};
\end{tikzpicture}
```

图 2.1.35 Node: 距离测算

发现没什么好办法获得中心距。

- 网格距

网格距使用 on grid 将全部距离都定位在网格上，以获得中心距。

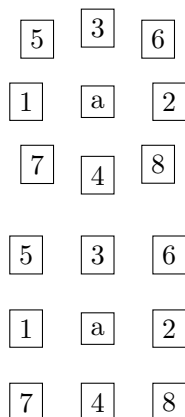


```
\begin{tikzpicture}[every node/.style = draw]
  \draw (0,0) grid (2,3);
  \node (a1) at (0,0) {normal};
  \node (a2) [above = 1 of a1] {1cm};
  \node (a3) [above = 1 of a2] {1cm};
  \begin{scope}[on grid]
    \node (b1) at (2,0) {grid};
    \node (b2) [above = 1 of b1] {1cm};
    \node (b3) [above = 1 of b2] {1cm};
  \end{scope}
\end{tikzpicture}
```

图 2.1.36 Node:on grid

- and 组合形式

上文已说过 and 可以添加多个修饰，这里来看一下用不用 and 的区别。

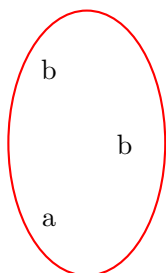


```
\begin{tikzpicture}[every node/.style = draw]
  \begin{scope}[node distance = 5mm]
    \node (a) at (2,3) {a};
    \node [left=of a] {1}; \node [right=of a] {2};
    \node [above=of a] {3}; \node [below=of a] {4};
    \node [above left=of a] {5}; \node [above right=of a] {6};
    \node [below left=of a] {7}; \node [below right=of a] {8};
  \end{scope}
  \begin{scope}[node distance = 5mm and 5mm]
    \node (a) at (2,0) {a};
    \node [left=of a] {1}; \node [right=of a] {2};
    \node [above=of a] {3}; \node [below=of a] {4};
    \node [above left=of a] {5}; \node [above right=of a] {6};
    \node [below left=of a] {7}; \node [below right=of a] {8};
  \end{scope}
\end{tikzpicture}
```

图 2.1.37 Node:and 详解

1.3.3 节点集

使用 fit 修饰可以将指定的节点圈起来。需要使用 fit 包



```
\begin{tikzpicture}
  \node (a) at (0,0) {a};
  \node (b) at (1,1) {b};
  \node (c) at (0,2) {c};
  \node[draw=red,inner sep=0pt,thick,ellipse,fit=(a) (b) (c)]
  {};
\end{tikzpicture}
```

图 2.1.38 Node:fit

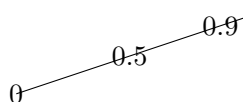
1.4 节点交互

1.4.1 曲线上的节点

节点交互指节点与其他 TikZ 图形之间的组合，例如在线段上某一位置插入节点。

- 节点插入位置

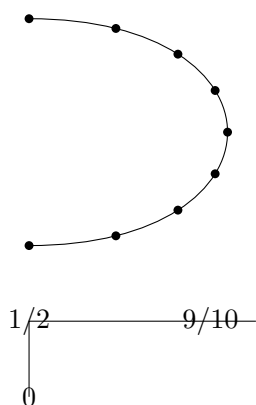
插入位置: [pos = <fraction>], 按线段长度比插入到对应的位置。



```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) -- (3,1) node [pos = 0] {0} node [pos = 0.5]
  {0.5} node [pos = 0.9] {0.9};
\end{tikzpicture}
```

图 2.1.39 Node: 节点插入位置

值得注意的是, pos 在曲线中代表的并不是长度比, pos 的具体计算方式比较复杂在这里不做说明。

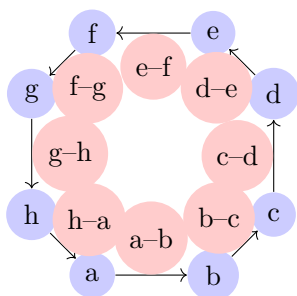


```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) .. controls +(right:3.5cm) and + (right:3.5cm)
    .. (0,3) node foreach \x in {0,0.125,...,1} [pos = \x,
      scale =3,radius = 1pt] {.};
  \draw (0,-2) |- (3,-1) node[pos=0]{0} node[pos=0.5]{1/2}
    node[pos=0.9]{9/10};
\end{tikzpicture}
```

图 2.1.40 Node:pos 位置

- 节点自动位置

自动位置: [auto = <direction>] 这个修饰将节点自动生成在线段的某一方位, 如果设置了 auto 的值, 则为 auto 方向, 如果没有设置, 则为最近一个设置的方向, 如果设置了 none, 则禁用 auto。其具体的规则请查阅 TikZ 官方文档。



```
\begin{tikzpicture}[scale=.8,auto=left,every node/.style={
  circle,fill=blue!20}]
  \node (a) at (-1,-2) {a};
  \node (b) at ( 1,-2) {b};
  \node (c) at ( 2,-1) {c};
  \node (d) at ( 2, 1) {d};
  \node (e) at ( 1, 2) {e};
  \node (f) at (-1, 2) {f};
  \node (g) at (-2, 1) {g};
  \node (h) at (-2,-1) {h};
  \foreach \from/\to in {a/b,b/c,c/d,d/e,e/f,f/g,g/h,h/a}
    \draw [->] (\from) -- (\to)
      node[midway,fill=red!20] {\from--\to};
\end{tikzpicture}
```

图 2.1.41 Node:auto

- 方位交换

交换: [swap]: 与原方向相反, 常常配合 auto 使用。需要用的 automate 包。swap 可以用' 来代替。

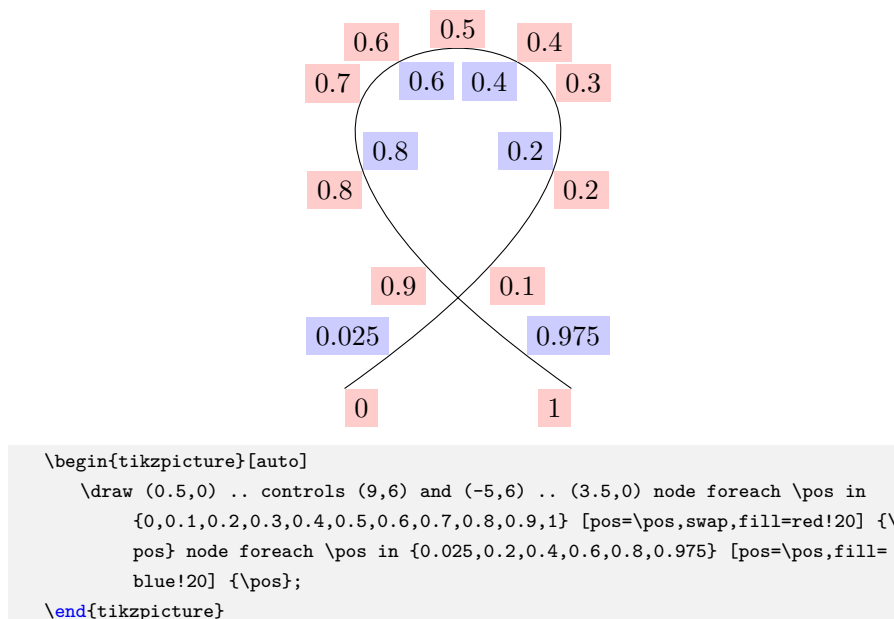
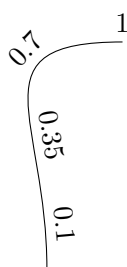


图 2.1.42 Node:swap

- 倾斜

倾斜: [sloped]: 倾斜可以让文本和曲线保持统一方向

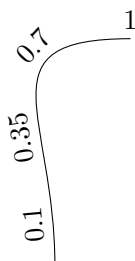


```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) .. controls +(up:2cm) and + (left:2cm)
    .. (1,3) node foreach \p in
      {0.1,0.35,0.7,1} [pos = \p,sloped,above] {\p}
    ;
\end{tikzpicture}
```

图 2.1.43 Node:sloped

- 颠倒

颠倒: [allow upside down]: 强制节点按指定方向生成, 这可能导致文字出现奇怪的朝向。

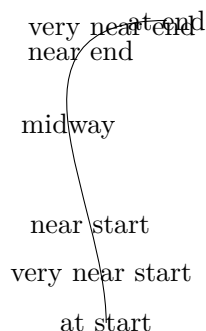


```
\begin{tikzpicture}[scale = 1]
  \draw (0,0) .. controls +(up:2cm) and + (left:2cm)
    .. (1,3) node foreach \p in
      {0.1,0.35,0.7,1} [pos = \p,sloped,above,
        allow upside down] {\p};
\end{tikzpicture}
```

图 2.1.44 Node:sloped

- 预定义位置

TikZ 预定义了几个 pos 对应的位置，可以直接使用其名称。



```
\begin{tikzpicture}[scale = 0.8]
  \draw (0,0) .. controls +(up:2cm) and +(left:3cm) .. (1,5)
    node[at end] {at end}
    node[very near end] {very near end}
    node[near end] {near end}
    node[midway] {midway}
    node[near start] {near start}
    node[very near start] {very near start}
    node[at start] {at start};
\end{tikzpicture}
```

图 2.1.45 Node: 预定义位置

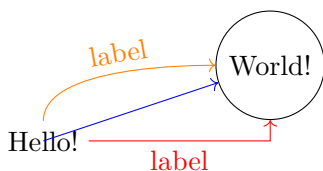
预定义位置具体位置如下表：

表 2.1 Node: 预定义位置

位置	对应 pos 值	位置	对应 pos 值
at start	0	very near start	0.125
near start	0.25	midway	0.5
near end	0.75	very near end	0.875
at end	1		

1.4.2 节点之间交互

节点之间除了默认的连线位置外，还可以使用 `node.<anchor>` 指定连接的位置。此外，与曲线类似，节点之间的连线也有多种样式。



```
\begin{tikzpicture}[scale = 1]
  \path (0,0) node (x) {Hello!}
    (3,1) node[circle,draw] (y) {World!};
  \draw [->,blue] (x.center) -- (y);
  \draw [->,red] (x) -| node[near start,below] {label} (y);
  \draw [->,orange] (x) .. controls +(up:1cm) and +(left:1cm)
    .. node[above,sloped] {label} (y);
\end{tikzpicture}
```

图 2.1.46 Node: 节点之间连线

1.4.3 节点图像

这里说明一些节点与图像之间的操作，部分操作只说明，但不写例子（写例子将改变全局图像设置），这些操作不是很常用，而且并不是所有的引擎都支持。

- 记录图像

```
\tikzset{every picture/.append style={remember picture}}
```

记录图像：[remember picture]：记录当前页所有图像位置等信息，这将在 `aux` 文件中增加对图像信息记录的数据，并不是所有引擎均支持，如果支持，需要编译两次。

- 覆盖

覆盖：[overlay = <boolean>]

启用此项，将可以使用其他图像中的节点，前提是对应的图像需要启用 `remember picture`，因此也需要编译两次



```
\tikz[remember picture] \node [circle,fill=red!50] (n1) {};
\tikz[remember picture] \node [circle,fill=blue!50] (n2) {};
\tikz[remember picture,overlay] \draw[->] (n1)--(n2);
```

图 2.1.47 Node:overlay

- 当前页

TikZ 提供了一个特殊的节点 `[current page]`，这可以在整个页面中进行节点操作。

```
\tikz[remember picture,overlay] \draw [line width=1mm,opacity = 0.25] (current page.center) circle (3cm);
```

图 2.1.48 Node:current page

1.5 节点拓展

节点拓展包含标签 (label)，大头针 (pin)，边缘 (edge)。他们都是基础写法的一种拓展，以不同的方式达到前文提到过的效果，可作为一种备选项。

1.5.1 节点标签

有时候我们需要在节点的周围添加一些文字信息，如果直接通过添加文字会让节点变大，如果再画一个纯文本的节点，又会消耗许多时间，这个时候就可以使用标签功能。标签的许多修饰与节点一样，不做过多解释。

- 标签

标签: `[label = <text>]`: 给予文本标签，默认在上方生成



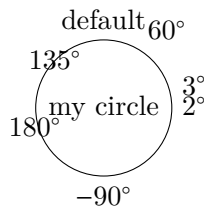
```
\begin{tikzpicture}[circle]
  \node [draw] (s) [label=$s$] {};
  \node [draw] (a) [right-of s] {} edge (s);
  \node [draw] (b) [right-of a] {} edge (a);
  \node [draw] (t) [right-of b, label=$t$] {} edge (b);
\end{tikzpicture}
```

图 2.1.49 Node:label

- 标签选项

标签选项: `label = {[<options>]<angle>:<text>}`，区别于主节点 (main node)，这里将标签称为标签节点 (label node)，与节点类似的，标签节点也可以使用 `every [] label/.style` 指定默认样式，标签节点参数含义如下。

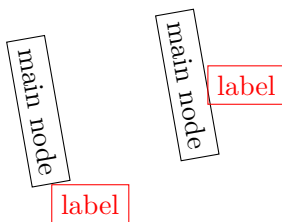
- 位置: `[position = <angle>]` (默认: above)
- 这里的 `<angle>` 与 `<offset>` 相同，但也可以设置具体的角度值。



```
\begin{tikzpicture}[scale = 1]
  \node [circle, draw,
    label=default,
    label=60:$60^\circ\textcolor{blue}{\circ}$,
    label=below:$-90^\circ\textcolor{blue}{\circ}$,
    label=3:$3^\circ\textcolor{blue}{\circ}$,
    label=2:$2^\circ\textcolor{blue}{\circ}$,
    label={[below]180:$180^\circ\textcolor{blue}{\circ}$},
    label={[centered]135:$135^\circ\textcolor{blue}{\circ}$}] {my circle};
\end{tikzpicture}
```

图 2.1.50 Node-Label:angle

- 绝对位置: [absolute = <boolean>] (默认: true)
若启用, 采用全局坐标, 否则, 采用主节点的坐标。



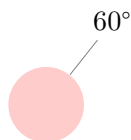
```
\begin{tikzpicture}[rotate=-80, every label/.style={draw, red}]
  \node [transform shape, rectangle, draw, label=right:label] at
    (0,0) {main node};
  \node [transform shape, rectangle, draw, label={[absolute]
    right:label}] at (0,2) {main node};
\end{tikzpicture}
```

图 2.1.51 Node-Label:absolute

1.5.2 大头针

大头针 pin = [options]<angle>:<text>: pin 和 label 用法几乎一致, 唯一的区别是 pin 会带上与主节点之间的连线。下面只对连线进行说明, 其他参考前文。

- 基本用法

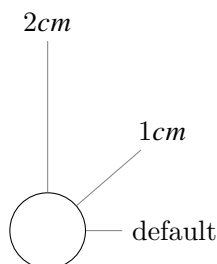


```
\begin{tikzpicture}[scale = 1]
  \node [circle, fill=red!20, minimum size = 1cm, pin = 60:$60^\circ\textcolor{blue}{\circ}$] {};
\end{tikzpicture}
```

图 2.1.52 Node-Pin: 基础用法

- 距离

大头针节点距: [pin distance = <distance>]

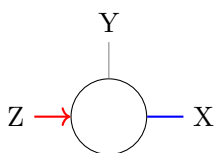


```
\begin{tikzpicture}[scale = 1]
  \node [circle, draw, minimum size = 1cm,
    pin = right:default,
    pin = {[pin distance = 1cm]above right:$1cm$},
    pin = {[pin distance = 2cm]above:$2cm$}] {};
\end{tikzpicture}
```

图 2.1.53 Node-Pin:distance

- 连线样式

连线样式: [pin edge = <options>]

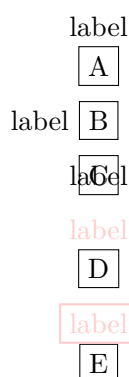


```
\begin{tikzpicture}[scale = 1]
  \node [circle,draw,minimum size = 1cm,
    pin={[pin edge={blue,thick}]right:X},
    pin={[pin edge={<-,red,thick}]left:Z},
    pin = above:Y] {};
\end{tikzpicture}
```

图 2.1.54

- 简化语法

Label 和 Pin 的语法有点复杂,可以启用 quotes 包,简化语法。与原来的 [options]<angle>:<text> 对应,新的语法形式为"<text>" [options]。



```
\begin{tikzpicture}[scale = 1]
  \matrix [row sep = 2mm] {
    \node [draw,"label"] {A}; \\
    \node [draw,"label" left] {B}; \\
    \node [draw,"label" centered] {C}; \\
    \node [draw,"label" color = red!20] {D}; \\
    \node [draw,"label" {red!20,draw,thick}] {E}; \\
  };
\end{tikzpicture}
```

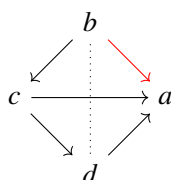
图 2.1.55 Node-quotes: 基础用法

1.5.3 边缘

边缘 (edge) 是节点间连线的一种方案,与一般连线不同的是,边缘连线会在所有连线绘制完成后再绘制,同样可以使用 every edge/.style 控制全部边缘样式。

- 边缘基础用法

边缘既可以在节点绘制过程中使用,也可以单独绘制。



```
\begin{tikzpicture}[scale = 1]
  \node (a) at (0:1) {$a$};
  \node (b) at (90:1) {$b$} edge [->] (a);
  \node (c) at (180:1) {$c$} edge [->] (a) edge [->] (b);
  \node (d) at (270:1) {$d$} edge [->] (a) edge [dotted] (b)
    edge [->] (c);
  % 这种写法效果一样
  \node foreach \name/\angle in {a/0,b/90,c/180,d/270}
    (\name) at (\angle:1) {$\name$};
  \path[->] (b) edge (a) edge (c) edge [-,dotted] (d)
    (c) edge (a) edge (d)
    (d) edge (a);
\end{tikzpicture}
```

图 2.1.56 Node-Edge: 基础用法

- 边缘在 quote 中的写法
和其他拓展方法相同，边缘也可以利用 quote 包。

$$\frac{\text{start} \quad \text{left}}{\text{right} \quad \text{end}}$$

```
\draw (0,0) edge ["left", "right", "start" near
start, "end" near end] (4,0);
```

图 2.1.57 Node:

III 百科

一、参数百科

参数¹在这里指 TikZ 中一些可选项 (option) 对应的通用值。

1.1 通用参数

通用参数，即常见的参数，其参数名和值是大部分同类语言都具备的，只需理解参数意思就可明白如何写值。

1.1.1 常见通用参数意义

表 3.1 常见通用参数意义

参数	值	意义	举例
dimension	数字	一般为长度，可自定义单位	line width = <dimension>
angle	数字	旋转角度，单位为度	rotate = <angle>
scaling	数字	缩放比例	scale = <scaling >

1.2 TikZ 特有参数

特有参数，即参数和对应的值是 TikZ 所特定的，TikZ 专有这些值，参数的值往往需要查表得知。

1.2.1 对齐: alignment option

表 3.2 参数: alignment option

值	意义	值	意义
left	左对齐	flush left	左对齐 (禁止拆分单词)
right	右对齐	flush right	右对齐 (禁止拆分单词)
center	居中对齐	flush center	居中对齐 (禁止拆分单词)
justify	拉长	none	禁止之前的对齐参数

¹这里借用了 css 的概念

1.2.2 锚点: anchor name

锚点主要用在定位与文本对齐上，锚点的值需要视具体的修饰选择。

表 3.3 锚点:anchor name

类型	值	意义	值	意义
对齐	center mid	文字中心对齐 ¹ 文本中心对齐 ³	base	文字底部对齐 ²
位置	north east north-east north-west	北 东 东北 西北	south west south-east south-west	南 西 东南 西南
组合	base west mid west	采用 base 对齐的西方 采用 mid 对齐的西方	base east mid east	采用 base 对齐的东方 采用 mid 对齐的东方

¹ 单个文字的中心。

² 文字底边对齐，英文四线三格中第三条线对齐。

³ 文本整体的中心。

1.2.3 偏移: offset

表 3.4 偏移:offset

值	意义	值	意义
above/below	上/下	left/right	左/右
above left/right	上左/右	below left/right	下左/右
centered	中心		