Asymptote 范例教程

刘海洋

2009 年 10 月 11 日

本文是基于 Asymptote 1.88 的一组教程。
http://asy4cn.googlecode.com/

目录

第一章　赵爽的弦图

赵爽博士是位老知识分子，研究兴趣是天文历法与算学，一生精研《周髀算经》。不过赵老爷子年轻的时候书籍都是手写，最近才紧跟时代潮流用上了电脑。现在他要修订他研究《周髀算经》的札记，决定使用 LaTeX 来排版。

现在他遇到一个难题，就是他要画出笔记中讲解勾股定理的一幅弦图。听人介绍，几经比较之后，他决定使用现在炒得火热的 Asymptote。

赵博士的原图是手画的，线框多有不直不准的，图 1.1 就是旧年据手稿做的雕版图，赵博士并不满意。赵爽博士理想中的图，线条要平整美观，文字要清楚整齐，图形还要上色：朱实自然得用红色，黄实也该用黄色，以与注文一致——就是图 1.2 的样子。

计议已定，赵博士要开始正式的绘图了。

1.1　绘图环境

Asymptote 的安装并不复杂，在 Windows 下面就是下载运行那个安装包，在 Linux 下面一般也只需要下载对应的压缩包，解压就可以使用了。哦，赵博士用的就是 Windows。

点图标运行 Asymptote，就出现了交互式的命令行，提示符是一个 >。输入命令：

```
draw((0,0) -- (3cm,4cm));   // 一条直线
```

赵博士装的 GSView 立即弹了出来，里面已经画出一条倾斜的直线。再输入 quit，程序退出，并留下了一个叫做 out.eps 的图形文件，小菜一碟。

这里稍稍解释一下上面的一句代码。draw 是画线的命令，更准确地说，是 Asymptote 中的函数：它带有一个参数 (0,0) -- (3cm,4cm)，参数外面是圆括号，整个命令以分号结束。里面的 (0,0) -- (3cm,4cm) 是由两个

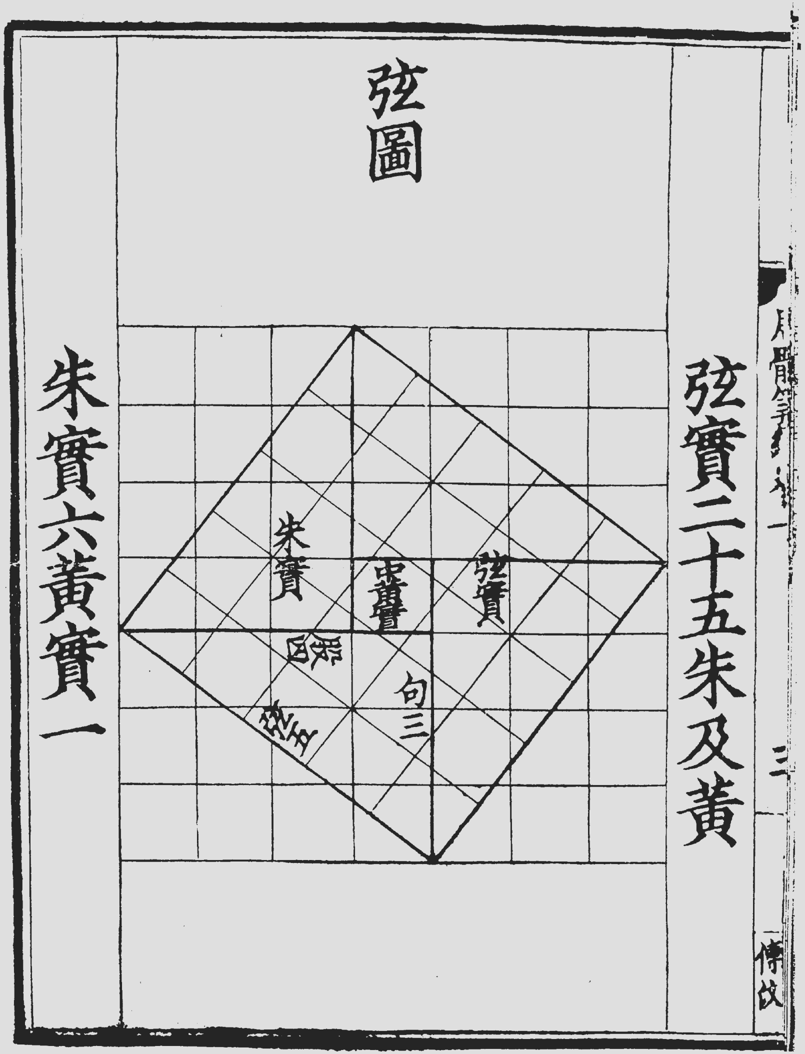


图 1.1: 旧年做的雕版

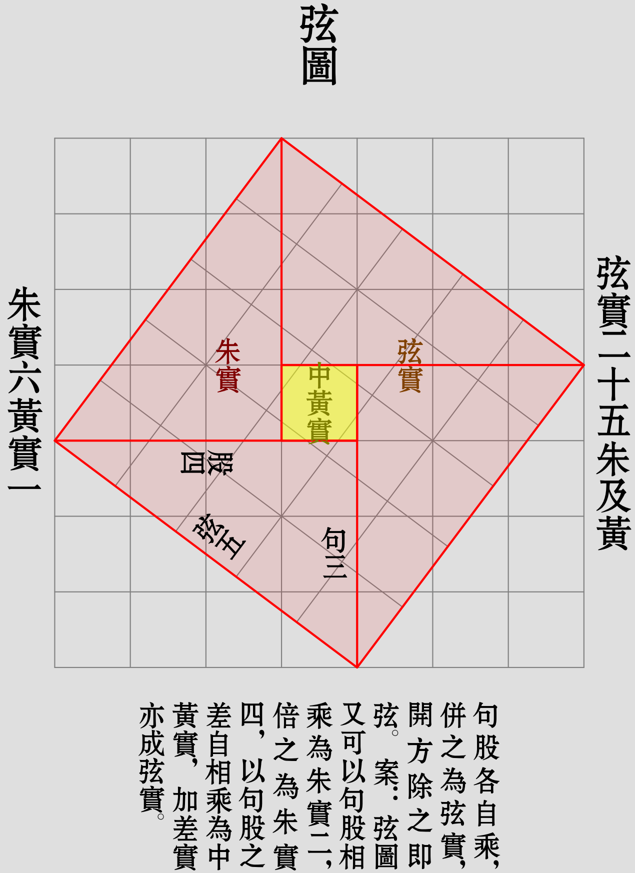

弦圖
朱實六黃實一
弦實二十五朱及黃
朱實
中黃實
弦實
股四
弦五
句三
句股各自乘，併之為弦實，開方除之即弦。案：弦圖又可以句股相乘為朱實二，倍之為朱實四，以句股之差自相乘為中黃實，加差實亦成弦實。

图 1.2: 理想中的新版本

坐标连接而成的直线，坐标是在直角坐标系下的，可以带单位 mm, cm, pt, bp, inch, inches，其意义与在TEX 中的一样，如果不带单位，则默认为 bp。行末以 // 开头的是注释，另有一种在 /* */ 之间的注释，与 C 语言相同。

不过赵博士写书讲求胸有成竹方才下笔，因此他更愿意使用更一种方式：打开一个文本编辑器，把上面的绘图命令都录入完毕，保存为一个名为 line.asy 的文件，最后把这个文件拖动到 Asymptote 的图标上面，就完成了整个作图。

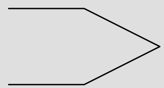
赵博士的小侄子不屑于这种快捷图标的编译方式，他直接进入命令行，输入 asy，就进入了 Asymptote 的交互环境；要是输入 asy line，就画出了刚才的保存的直线。

1.2 直线与绘图命令

弦图的图形其实很简单，都是直线、方块、三角形这些，而且为了计算的简便，所有长度也都是整数值。那么，首先就是在 Asymptote 中来画直线。

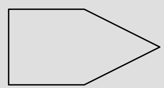
正如前面试验的时候做的那样，一条直线就是用 -- 把坐标连起来，再使用 draw 命令，直线就画出来了。事实上，可以用 -- 把坐标点连成折线：

```
draw( (0,0) -- (1cm,0) -- (2cm,0.5cm) -- (1cm,1cm) -- (0,1cm) );
```



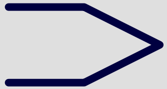
像这样把坐标用 -- 连结起来的，就成为一条路径。把直线而稍做修改 ，在后面连上一个特殊的坐标 **cycle**，就可以得到一条首尾相接的闭路径。如：

```
draw( (0,0) -- (1cm,0) -- (2cm,0.5cm) -- (1cm,1cm) -- (0,1cm) -- cycle );
```



画一条路径可以使用不同的颜色、粗细的笔，这只要给 draw 命令多加一个画笔 参数（多个参数用逗号分开）：

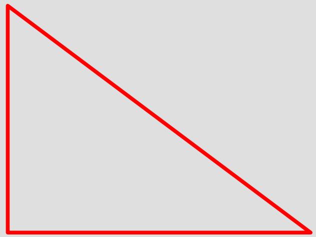
```
draw((0,0) -- (1cm,0) -- (2cm,0.5cm) -- (1cm,1cm) -- (0,1cm), darkblue+1mm);
```



这里 `darkblue` 是颜色，`1mm` 是线的粗细。`darkblue+1mm` 即指一毫米宽的深蓝色粗线。（可用的颜色名称可以参考 [1]）

赵博士画的是“勾三股四弦五”的红色三角形，这很容易：

```
draw( (0,0) -- (4cm,0) -- (0,3cm) -- cycle, red+0.5mm );
```

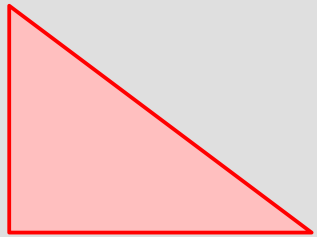


不过现在还需要的是实心的三角形，因此就需要一个新的绘图命令 `fill`，即填充。于是，红色三角形就成为

```
fill( (0,0) -- (4cm,0) -- (0,3cm) -- cycle, red );
```

但这样一来三角形的颜色就太重了，而且边界也不清楚。因此似乎应该先用浅红色填充一遍，然后再用深红色勾边。好在可以使用一个命令 filldraw 同时完成这两件事情，这样就不需要把一条路径写两遍了。即有：

```
filldraw((0,0) -- (4cm,0) -- (0,3cm) -- cycle,
    fillpen=palered, drawpen=red+0.5mm);
```



这里可以简单地直接写两个参数 palered, red+0.5mm，不过为了清晰起见还是使用 “键 = 值” 的写法，明确表示出填充的画笔和描线的画笔。

现在，赵博士的整个弦图的框架就呼之欲出了，就是画出四个三角形（图 1.3）：

```
filldraw( (4cm,0) -- (4cm,3cm) -- (0,3cm) -- cycle,
    fillpen=palered, drawpen=red+0.5mm);
filldraw( (7cm,4cm) -- (4cm,4cm) -- (4cm,0) -- cycle,
    fillpen=palered, drawpen=red+0.5mm);
filldraw( (3cm,7cm) -- (3cm,4cm) -- (7cm,4cm) -- cycle,
    fillpen=palered, drawpen=red+0.5mm);
filldraw( (0,3cm) -- (3cm,3cm) -- (3cm,7cm) -- cycle,
    fillpen=palered, drawpen=red+0.5mm);
```

还应该画出弦图的中间的“黄实”，用黄色填充。这部分是一个正方形，可以使用现成的 box(角点, 角点) 命令来产生矩形的路径，因而填充正中间的正方形就可以用：

```
fill( box((3cm,3cm), (4cm,4cm)), yellow );
```

这个填充的命令应该放在画线之前（以免覆盖描的红线）。

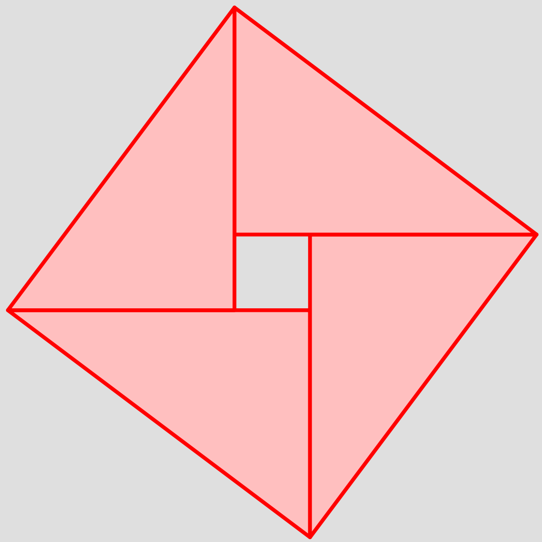


图 1.3: 弦图的初步框架

回顾前面的代码，赵博士觉得连续地写四个 filldraw 命令太重复了。他读了 Asymptote 的文档 [1]，才知道多条路径可以用符号 ^^ 连起来，一起使用，于是立即着手改进原来的代码。

而且，由于还打算在图的后面画出参考网格，图形的颜色还应该设置为半透明的。好在这并不难实现，只要稍稍改动一下填充的画笔，使用 opacity(数值) 来设定有一定不透明度（取值为 0 ∼ 1）的画笔，并把它加在原来的画笔上。于是赵博士最后写出了这样的代码：

```
fill( box((3cm,3cm), (4cm,4cm)), opacity(0.5)+yellow );
filldraw( (4cm,0) -- (4cm,3cm) -- (0,3cm) -- cycle
    ^^ (7cm,4cm) -- (4cm,4cm) -- (4cm,0) -- cycle
    ^^ (3cm,7cm) -- (3cm,4cm) -- (7cm,4cm) -- cycle
    ^^ (0,3cm) -- (3cm,3cm) -- (3cm,7cm) -- cycle,
    fillpen=opacity(0.1)+red, drawpen=red+0.5mm );
```

至此，弦图的主要框架（图 1.4）就此完成。

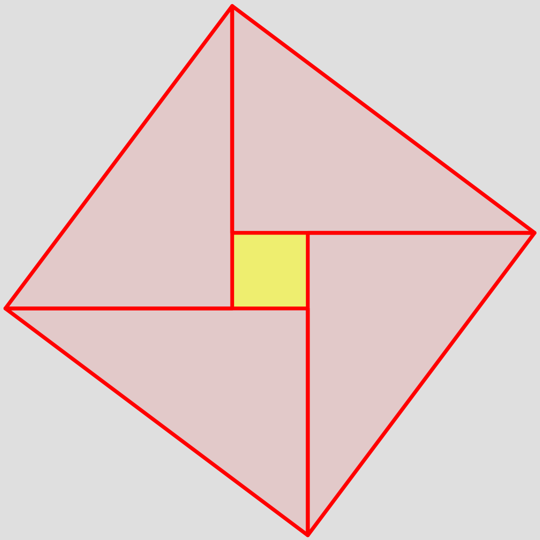


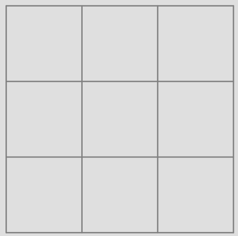
图 1.4: 弦图的进一步优化的框架

1.3 图形变换与功能模块的使用

然后是画作为长度参考的网格。其实网格一开始就该在图中画出来，这样后面画图准确与否才能看得清楚。不过现在赵博士是重作旧图，图样已定，网格就搁在弦图的主要图形后面才画了。

按说这个网格是十分简单的，无非就是画一些灰色的纵横细线。比如这个 3×3 的网格：

```
draw( (0,0) -- (3cm,0)
    ^^ (0,1cm) -- (3cm,1cm)
    ^^ (0,2cm) -- (3cm,2cm)
    ^^ (0,3cm) -- (3cm,3cm),
    gray );
draw( (0,0) -- (0,3cm)
    ^^ (1cm,0) -- (1cm,3cm)
    ^^ (2cm,0) -- (2cm,3cm)
    ^^ (3cm,0) -- (3cm,3cm),
    gray );
```



可无疑这个办法显得太麻烦了，赵博士要画的是 7×7 的网格，就要分别画出 16 条直线。这样的代码不仅不好写，而且容易出错，修改一下也很麻烦。

赵博士看到了手册 [1] 中讲循环语句的用法，似乎可以完成这件事。可是以赵博士的年龄，再去看什么编程什么变量的，命令不能一条一条执行下来，很不习惯，头脑就往往转不清楚。

于是赵博士就去论坛上咨询，一些人劝他去用几行循环语句，甚至有人已经把完整的函数做好了。但有一个结

果特别引人注目，有人指出，在 math 模块中已经定义好了一个 grid 函数，只要拿来用就可以了。赵博士立即精神大振，来看这个 grid 函数：

```
picture grid(int Nx, int Ny, pen p=currentpen)
```

这个是在 math 模块中 grid 函数的原型。它说明 grid 函数有 Nx, Ny 两个整数类型的必需参数，一个可选的画笔，并且返回一个 **picture**（图）类型的对象。

要使用模块的功能，需要在绘图之前导入这个模块，这只要使用

```
import 模块名;
```

因此，要使用 math 模块中的 grid 函数，只要在代码中写

```
import math;
```

就可以了。

grid 函数的行为看起来很奇怪，调用它会在一个单独的图上画出一个 Nx × Ny 的网格，网格的左下角在原点，间距为 1。要使用 grid 函数画的图形，要使用 add(图) 命令，把这个图形加在当前的图上：

```
import math;
add(grid(10,10,gray));
```

不过直接这样做的结果是只能得到一个小得已经看不清的网格。因此，必须对图形进行放缩。

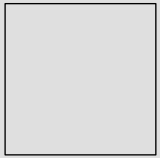
Asymptote 提供了平移、旋转、放缩、倾斜、反射等各种的仿射变换，来对坐标、路径、图形等元素进行变换（严格的函数原型参考 [1]）：

```
shift(坐标)          // 按坐标平移
shift(x, y)          // 按 (x, y) 平移
scale(倍数)          // 按倍数放缩
xscale(倍数)         // x 轴方向按倍数放缩
yscale(倍数)         // y 轴方向按倍数放缩
scale(x, y)          // 在 x 轴、y 轴方向分别按倍数 x, y 放缩
```

```
rotate(角度, z=(0,0)) // 按角度绕中心 z（默认为原点，逆时针）旋转
slant(因子)           // 按一定因子向右倾斜
reflect(a, b)         // 相对直线 a--b 反射
```

使用一个变换就是把这个变换乘在被变换对象的左边。例如一个放缩一个单位正方形：

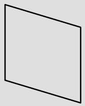
```
draw( scale(2cm) * box((0,0), (1,1)) );
```



这样的变换可以连续地做下去，例如

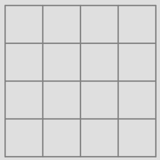
```
draw( rotate(90) * slant(0.3) * scale(1cm) * box((0,0), (1,1)) );
```

就是把一个单位正方形先放大，再倾斜，再旋转 $90°$：



有了这些变换，画出一个合适大小的网格就不再是什么难事了：

```
import math;
add( scale(5mm) * grid(4, 4, gray) );
```



赵博士的弦图有两个网格，不仅要大小合适，而且其中一个需要进行旋转和平移。平移的位置很明显，但旋转仍然需要一些计算。当然这难不倒精研天文算学多年的赵博士，这里弦实（5×5 的正方形）可以看作是顺时针旋转得到的，从朱实的三角形容易看出旋转的角度正好是 $\arctan(3/4)$。Asymptote 中也可以方便地调用返回角度的反三角函数 aTan 来计算这个角度。更详细的数学函数列表，参看 [1]。

于是，赵博士弦图中的网格，就可以这样方便地画出来了（图 1.5）：

```
// 网格
import math;
add( scale(1cm) * grid(7, 7, gray) );
add( shift(0,3cm) * rotate(-aTan(3/4)) * scale(1cm) * grid(5, 5, gray) );
// 弦图主体
fill( box((3cm,3cm), (4cm,4cm)), opacity(0.5)+yellow );
filldraw( (4cm,0) -- (4cm,3cm) -- (0,3cm) -- cycle
    ^^ (7cm,4cm) -- (4cm,4cm) -- (4cm,0) -- cycle
    ^^ (3cm,7cm) -- (3cm,4cm) -- (7cm,4cm) -- cycle
    ^^ (0,3cm) -- (3cm,3cm) -- (3cm,7cm) -- cycle,
    fillpen=opacity(0.1)+red, drawpen=red+0.5mm );
```

1.4 标注文字

现在要进行的是文字的标注。按照勾股定理的约定，赵博士打算在一个红色三角形内标注“朱实”，在黄色矩形处标注“黄实”，并为拼得的整个大矩形标注“弦实”；在另一红色三角形的三边标注“勾三”、“股四”、“弦五”的尺寸；最后在图形两侧加上说明的文字。

在标注文字之前，对于中文标签，应该先定义好中文环境和字体。Asymptote 会调用 LaTeX 来进行标签的处理，因而需要设置的就是 LaTeX 的编译引擎与一般的中文 LaTeX 文件导言区。在这里，赵博士决定使用 XeTeX 引擎与 xeCJK 宏包来处理中文。为此，在 Asymptote 源文件中，他使用了下面的设置代码：

```
settings.tex = "xelatex";
usepackage("xeCJK");
```

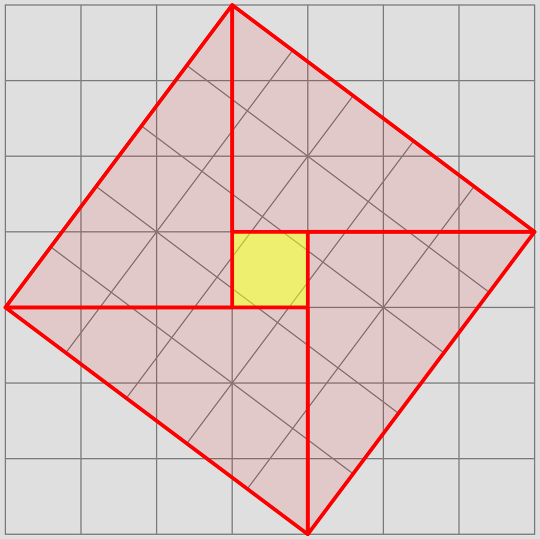


图 1.5: 带网格的弦图草图

```
texpreamble("\setCJKmainfont{SimSun}");
```

这里第一行是设置编译时所用的 TeX 引擎。后面 usepackage 命令就是 LaTeX 中的 **\usepackage** 命令的一个包装形式，里面的字符串参数就是宏包名；而 texpreamble 命令则把接收的参数直接放进 LaTeX 的导言区。

进行上述设置后，就可以正确使用中文标签了。标注的命令很简单，就是 label，参数正是标签文字和标签的位置。例如：

```
draw( (0,0) -- (1cm,1cm) -- (2cm,0) );
label( "中间", (1cm,0cm) );
```

就得到

可以在标签中使用任意的 LaTeX 代码，包括数学公式，例如：

```
label("$x = \sin\alpha$", (0,0));
```

就会正确地得到 $x = \sin\alpha$ 的标签。

现在，我们可以给弦图加上“朱实”、“黄实”和“弦实”的标签了。在前面的框架代码后面加上

```
label("朱实", (2cm,4cm));
label("黄实", (3.5cm,3.5cm));
label("弦实", (5cm,4cm));
```

以及设置字体的代码，就得到图 1.6 的结果。

下面则是要给三角形的三边进行标注。

与前面在一个点处标注不同，这里实际是给一条路径（直线）标注标签。因此，想要得到的是距离这条路径的中点一定方向距离加一个标签，而不是简单地取路径上的一点作为标签的位置。好在 Asymptote 确实也提供了这样的功能，仍然使用 label 命令，基本语法是：

```
label(标签, 路径)
```

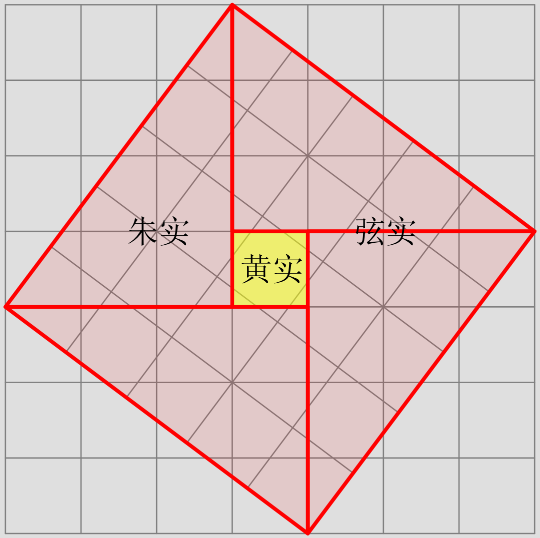
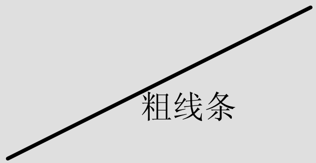


图 1.6: 带部分标注的弦图

这里默认会在路径中间的右侧（沿着路径行进方向）加标签。例如：

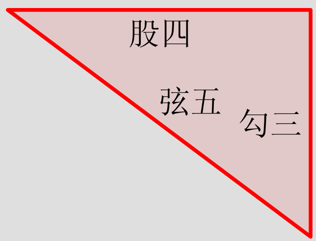
```
draw( (0,0) -- (4cm,2cm), linewidth(0.5mm) );
label("粗线条", (0,0) -- (4cm,2cm));
```

（其中的 linewidth 函数用来表示具有一定线宽的画笔）这段代码将得到



现在，赵博士就可以给三角形的三边加上“勾三”、“股四”、“弦五”的标签了，只要稍稍注意一下标签摆放的默认方向：

```
filldraw( (4cm,0) -- (4cm,3cm) -- (0,3cm) -- cycle,
    fillpen=opacity(0.1)+red, drawpen=red+0.5mm );
label( "勾三", (4cm,3cm) -- (4cm,0) );
label( "股四", (0,3cm) -- (4cm,3cm) );
label( "弦五", (4cm,0) -- (0,3cm) );
```

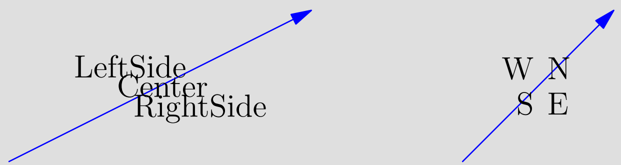


不过，为了得到正确的标签位置，不得不把原来逆时针画的线用顺时针方向重写，这多少让赵博士有些恼火：为什么不能在路径的左边标注标签呢？确实可以，很简单，只要给 label 命令再加上 align=LeftSide 选项（或

者简单地只用 LeftSide）就指定了在左边放置对齐。同理，还有向右对齐的 RightSide，在中间对齐的 Center 以及一般意义的相对方向 Relative(方向)。例如：

```
draw( (0,0) -- (4cm,2cm), blue, Arrow );
label( "LeftSide", (0,0) -- (4cm,2cm), align=LeftSide );
label( "RightSide", (0,0) -- (4cm,2cm), align=RightSide );
label( "Center", (0,0) -- (4cm,2cm), align=Center );

draw( (6cm,0)--(8cm,2cm), blue, Arrow );
label( "E", (6cm,0)--(8cm,2cm), Relative(E) );
label( "S", (6cm,0)--(8cm,2cm), Relative(S) );
label( "W", (6cm,0)--(8cm,2cm), Relative(W) );
label( "N", (6cm,0)--(8cm,2cm), Relative(N) );
```



E, S, W, N 分别是东南西北四个罗盘方向，用在 Relative 函数里面就表示相对于路径方向的四个方向。为明确，这里用 Arrow 选项在画线时加了箭头。

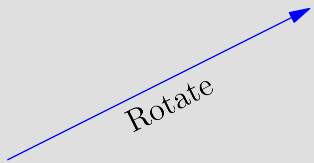
不过，这样的标签还是不能令赵博士满意。赵博士给三角形加标签，还希望标签随着三角形的边作旋转，使标签沿着边排列。为此，赵博士不得不又仔细查看了 [1]，在讲标注文字一节（label）他找到了一个构造标签的更高级的办法，即不仅仅是使用一个简单的字符串，而是使用

```
Label(标签)
```

函数进行构造。里面的“标签”参数仍然可以是原来的字符串，或是通过这个函数构造出来的高级的标签。这个函数可以带许多可选的其他参数，如 **position=位置** 的参数就可以指定标签放在路径中点之外的其他地方；而 embed=嵌入变换方式 的参数则可以解决标签自动旋转的问题。

这里暂且放下 **position** 参数。只来看 embed 参数的一个特例：Rotate(方向)。这个参数会让标签向着给定的方向旋转，如：

```
draw( (0,0)--(4cm,2cm), blue, Arrow );
label( Label("Rotate", Rotate((4,2))),
    (0,0)--(4cm,2cm) );
```



这里坐标 $(4,2)$ 正是这条直线的绘制方向。

终于，使用了上面的全部功能，赵博士完成了全部的图形标注工作（图 1.7）：

```
settings.tex = "xelatex";
usepackage("xeCJK");
texpreamble("\setCJKmainfont{SimSun}");

import math;
add( scale(1cm) * grid(7, 7, gray) );
add( shift(0,3cm) * rotate(-aTan(3/4)) * scale(1cm) * grid(5, 5, gray) );

fill( box((3cm,3cm), (4cm,4cm)), opacity(0.5)+yellow );
filldraw( (4cm,0) -- (4cm,3cm) -- (0,3cm) -- cycle
    ^^ (7cm,4cm) -- (4cm,4cm) -- (4cm,0) -- cycle
    ^^ (3cm,7cm) -- (3cm,4cm) -- (7cm,4cm) -- cycle
    ^^ (0,3cm) -- (3cm,3cm) -- (3cm,7cm) -- cycle,
    fillpen=opacity(0.1)+red, drawpen=red+0.5mm );
```

```
label("朱实", (2cm,4cm));
label("黄实", (3.5cm,3.5cm));
label("弦实", (5cm,4cm));
label( Label("勾三",Rotate(S)), (4cm,0)--(4cm,3cm), LeftSide );
label( Label("股四",Rotate(E)), (4cm,3cm)--(0,3cm), LeftSide );
label( Label("弦五",Rotate((4,-3))), (0,3cm)--(4cm,0), LeftSide );
```

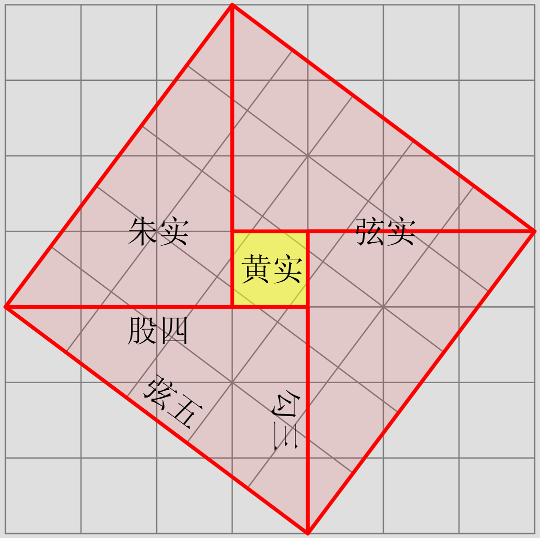


图 1.7: 带标注的完整弦图

习题

1. 查阅参考手册 [1]，看看都有哪些颜色可用。看看你的系统中安装了哪些中文字体。然后修改图 1.7，使用另一种你喜欢的字体进行标注；并且将“朱实”、“黄实”和“弦实”分别用红、黄、橙色标注。

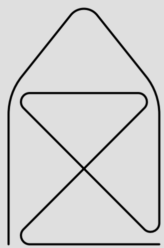
2. 查阅参考手册 [1]，看看 `draw`、`fill`、`label` 等命令都可以带哪些可选的参数，并自己举例子试验一下效果。
3. 代码

```
guide zhushi = (4cm,0) -- (4cm,3cm) -- (0,3cm) -- cycle;
```

   可以把一个三角形的路径保存在变量 `zhushi` 中，以后就可以使用 `draw(zhushi)` 这样的命令对此路径进行操作了[1]。

   考虑如何利用平移和旋转变换，只定义一个三角形的路径，就把弦图中的四个红色三角形都画出来。

4. 查阅手册 [1] 和 Asymptote 自带的例子，研究模块 roundedpath 的用法。并利用它尝试画出下面的图形：



   看看 Asymptote 中还有什么用法简单而又有趣的模块。

5. （较难）研究在 LaTeX 里中文直排的方法，尽量精确地复现出赵博士理想中的弦图效果（图 1.2）。

[1]这里定义了一个 **guide** 类型的变量，我们译之为“路向”，以区别于“路径”（**path**）。两个概念在 Asymptote 中同源而有别，但本章中二者可以互换并无区别，因此为方便我们也不区别而通称为路径。有关 Asymptote 语言中路径与路向的区别，以及关于变量定义和使用的详细内容，请参考后续的章节及手册 [1]。

第二章　André Deledicq 的铺砌插画

André 是一名兴趣广泛的法国数学教师，在他的新著《Le monde des pavages》(《铺砌世界》) 中，打算画一幅有关羊的铺砌插画：

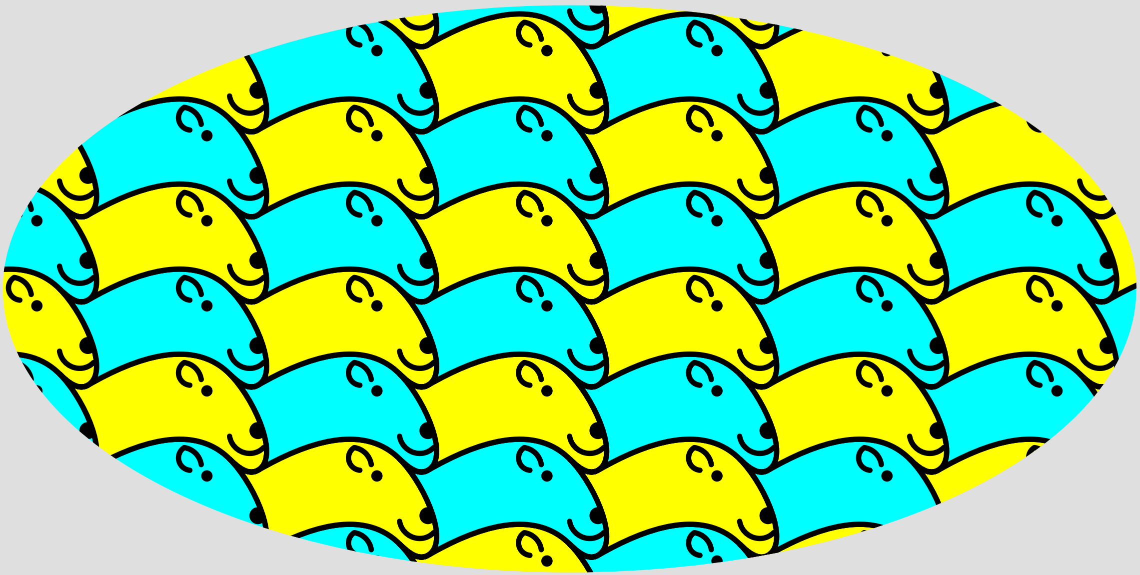
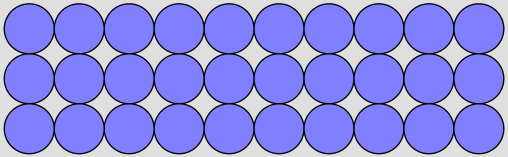


图 2.1: André 理想中的铺砌图

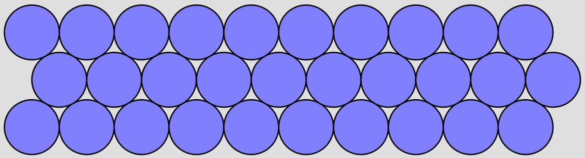
André 很清楚他要画的图形的数学理论，但 André 的朋友 Timothy 告诉他要画这样的图形多少是需要一些编程的知识的，对于他这样一位往日对计算机并不通晓的人来说可能会有困难。不过 André 并不以为意：这世上还有什么比数学更难的呢？于是他兴致勃勃的开始了。

2.1 从矩形到铺砌

铺砌图，顾名思义，就是像铺地板砖一样，把许多相同样式的图形平铺开来。不过，并不是什么图形都可以平铺填满整个平面的——比如圆形就不行。把许多圆形一个挨一个排列起来，也只能得到



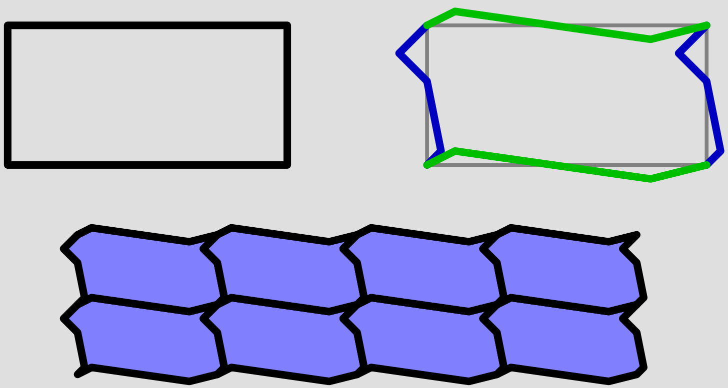
或者是



都会留下许多空隙。而矩形、平行四边形、六边形等等都可以不留空隙地把平面铺满。

但问题是，如何设计出 André 理想中的那种看起来形状不规则的铺砌图案呢？

身为数学教师的 André 当然有办法。其实不规则铺砌图案还是规则图案的变形。 André 要画的羊形铺砌图，其实就是从矩形铺砌变化而来的。只要把一个矩形图案的上下两边、左右两边分别变形，使得变形后的上边与下边、左边与右边还对应重合，就依然可以完美地拼合起来。这正是铺砌图案最基本的构成方式：



有了这个方法，对复杂的铺砌图，也只要从一个基本形状（比如矩形、正六边形）开始变形，就等到铺砌所需要的一块“砖”。

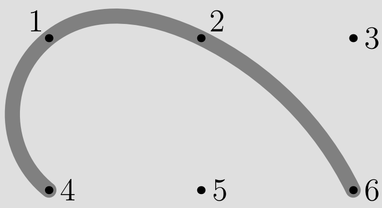
因此，要画出羊头形状铺砌图，只要把一个矩形按照上面的要求变形为一个羊头形状，在不同的位置重复画出就可以了。

2.2 变量与曲线

下面的问题就是，怎么画一个羊头呢？更具体地说，怎么画出羊头的曲线呢？

那么，首先要了解如何在 Asymptote 中描述曲线。1.2 节中提到 -- 连结一组坐标就成为直（折）线段；类似地，用 .. 连结坐标就得到经过这些坐标点的曲线：

```
size(5cm,0);
pair z1 = (0,1), z2 = (1,1), z3 = (2,1),
     z4 = (0,0), z5 = (1,0), z6 = (2,0);
guide p = z4 .. z1 .. z2 .. z6;
draw(p, gray+2mm);
```



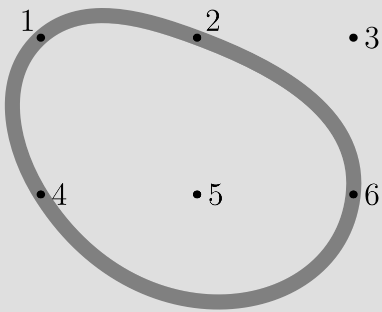
在这里，我们定义了一些变量以使代码清晰（这里略去了画点和标签的代码）。**pair** 类型的变量 z1, ⋯, z6 保存六个坐标，**guide** 类型的变量 p 保存一条曲线的路向。因而上面 size 之后的绘图代码就相当于

```
draw( (0,0) .. (0,1) .. (1,1) .. (2,0), gray+2mm );
```

其中前面的一句 size(5cm,0) 表示代码中的坐标只是相对位置，最后将整个图形按比例放缩为 5 cm 宽[1]。类似地，也可以使用 size(0,4cm) 把图形放缩到 4 cm 高。

最重要的当然还是曲线的表示。以 .. 连结的坐标会以一种尽量接近圆弧的方式连为经过这些点的光滑曲线。与画直线类似，**cycle** 可以作为一个特殊的坐标产生闭合曲线，即一条闭路向：

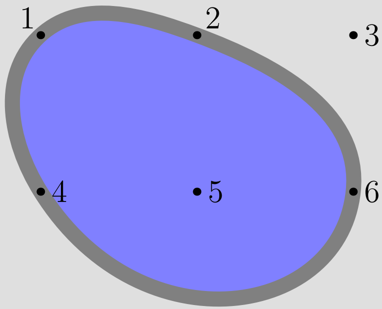
```
guide q = z4 .. z1 .. z2 .. z6 .. cycle;
draw(q, gray+2mm);
```



变量不仅仅是给了坐标、路向等对象一个简洁的名字，它也使得对同一个对象重复使用并进行不同的操作变得十分方便：

[1]注意坐标、图形会被放缩，但画笔的宽度不会放缩。

```
fill(q, lightblue);
draw(q, gray+2mm);
```

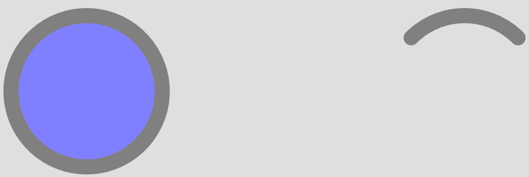


就像使用 box 可以直接得到矩形一样，最常用的曲线：圆、椭圆和圆弧，也可以使用现成的命令得到：

circle(c, r)	圆心 c，半径 r 的圆，这是逆时针方向的闭曲线；
ellipse(c, a, b)	中心为 c，长半轴 a，短半轴 b 的椭圆，这也是逆时针方向的闭曲线；
arc(c, r, angle1, angle2)	圆心 c，半径 r，角度从 angle1 到 angle2 的圆弧。

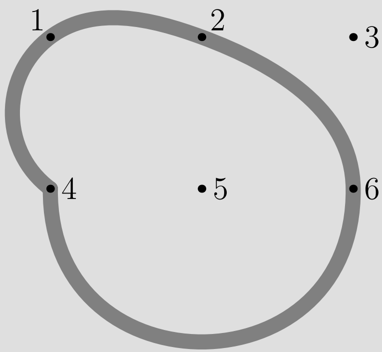
例如：

```
filldraw( circle((0,0), 1cm), lightblue, gray+2mm );
draw( arc((5cm,0), 1cm, 45, 135), gray+2mm );
```



一条用 **cycle** 产生的闭路向和简单地把首尾结点重合的路向是非常不同的。首先，只有闭路向可以填充颜色；其次，使用 **cycle** 连结的曲线在起点处是光滑连接的，而如果只是首尾结点重合则不会光滑连接。试将下面的曲线 q2 与上面的曲线 q 比较：

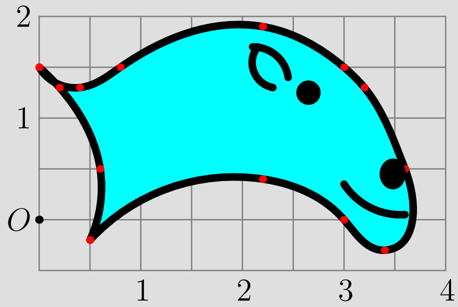
```
guide q2 = z4 .. z1 .. z2 .. z6 .. z4;
draw(q2, gray+2mm);
```



现在有了绘制曲线的方法，画出一个羊头就只是把草稿上的坐标连接起来而已。André 有一个纸上的草图，于是在描出几个点以后，他很快得到这样的结果（这里给图形增加了辅助网格）：

```
size(0,4cm);
pen outline = black+1mm;
// 头
guide head = (0.5,-0.2) .. (0.6,0.5) .. (0.2,1.3) .. (0,1.5) .. (0,1.5)
    .. (0.4,1.3) .. (0.8,1.5) .. (2.2,1.9) .. (3,1.5) .. (3.2,1.3)
    .. (3.6,0.5) .. (3.4,-0.3) .. (3,0) .. (2.2,0.4) .. (0.5,-0.2) .. cycle;
filldraw(head, cyan, outline);
dot(head, red+1mm); // 画出羊头曲线上的结点
// 五官
fill( circle((2.65,1.25), 0.12), outline );
```

```
fill( (3.5,0.3) .. (3.35,0.45) .. (3.5,0.6) .. (3.6,0.4) .. cycle, outline );
draw( (3,0.35) .. (3.3,0.1) .. (3.6,0.05), outline );
draw( (2.3,1.3) .. (2.1, 1.5) .. (2.15,1.7), outline );
draw( (2.1,1.7) .. (2.35,1.6) .. (2.45,1.4), outline );
```



在一开始，André 使用

```
pen outline = black+1mm;
```

定义一个 **pen** 类型的变量 `outline` 表示用来画羊头轮廓的画笔，以备使用。

然后，André 直接用 `..` 连结一组坐标来定义羊的头部轮廓：

```
guide head = (0.5,-0.2) .. (0.6,0.5) .. (0.2,1.3) .. (0,1.5) .. (0,1.5)
    .. (0.4,1.3) .. (0.8,1.5) .. (2.2,1.9) .. (3,1.5) .. (3.2,1.3)
    .. (3.6,0.5) .. (3.4,-0.3) .. (3,0) .. (2.2,0.4) .. (0.5,-0.2) .. cycle;
```

需要尖角的时候，就使用重复的相同点（如这里的起点）；曲线变化大的地方，取的点也比较密集。

最后五官的绘制。眼睛是填充的小黑圆，鼻子是黑色的卵形，耳朵和嘴都是简单的曲线。

于是，只要把这样一个图形一个挨一个地重复画许多遍，就可以得到 André 想要的铺砌效果了。设计羊头形状的工作无疑是最关键也最复杂的，因此 André 的任务现在就已经完成了一半。

不过继承了法国完美主义风气的 André 老师，很快挑出了毛病：这只羊头部的轮廓，并不完全是按照 2.1 节对矩形变形得到的——他的手稿基本上是这样设计的，但在使用 Asymptote 上绘图时则只是在手稿上相当随意地取了

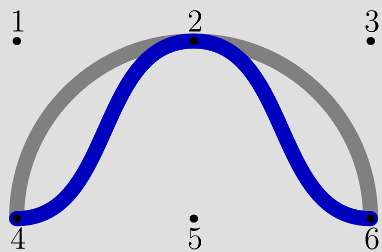
一些结点连结得到曲线，这个轮廓想必也并不能严丝合缝地一个个拼起来。还有一件很令他恼火的事情则是：要画出羊头的轮廓，他要画的点太多了，一个尖角用两个结点表示，也太不符合他的简洁美学了。因此，这个看上去相当不错的羊头一号，就被 André 老师无情地否决掉了。他决定发扬数学教师严谨简洁的作风，再做出更完美的羊头二号来。

2.3 细致的曲线调整与曲线操作

在 Asymptote 中，除了简单地使用 `--` 和 `..` 连接坐标来定义直线与曲线以外，也提供了更丰富的手段来对曲线进行更为细致的调整与操作。于是，就有了方向、张力、卷曲值这些东西。

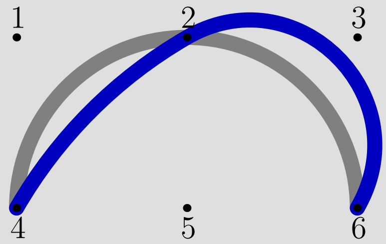
在曲线的结点处，可以使用以花括号括起的一个向量坐标来限定曲线在此处的切线方向 。例如：

```
draw(z4 .. z2 .. z6, gray+2mm);
draw(z4{right} .. z2{right} .. z6{right}, heavyblue+2mm);
```



在这里，常量 `right` 就相当于罗盘方向 `E`，也就是 `(0,1)`。事实上，在 Asymptote 中已经预定义好了上下左右四个方向：up, down, `left`, `right`，作为四个罗盘方向的同义词。此外，也可以使用预置的函数 `dir(角度)` 来表示指向此方向的单位向量，例如：

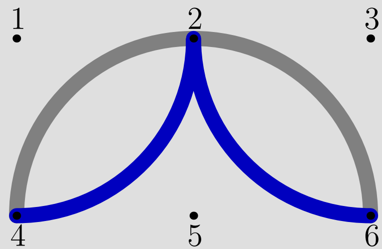
```
draw(z4 .. z2 .. z6, gray+2mm);
draw(z4{dir(60)} .. z2{dir(30)} .. z6{dir(-120)}, heavyblue+2mm);
```



当然，这里并不要求使用单位向量来限定曲线方向，使用任何长度的向量都得得到相同的效果。

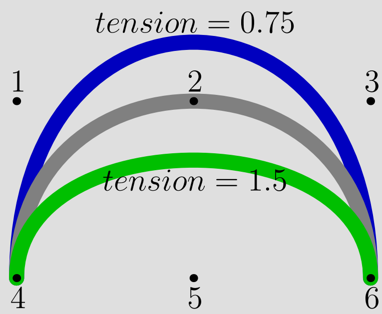
就像上面的例子，限定曲线的切线方向无疑使得定义一条曲线更加直观和方便。不仅如此，我们甚至还可以在曲线同一结点的两侧设置截然不同的切线方向，此时曲线就会在这个结点的位置得到一个尖角。如：

```
draw(z4 .. z2 .. z6, gray+2mm);
draw(z4 .. {up} z2 {down} .. z6, heavyblue+2mm);
```



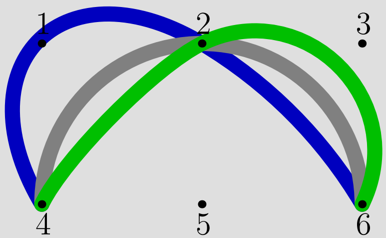
了解了曲线的方向，再来看曲线的张力。大体上讲，这是一个相对模糊的概念，因为它并不像曲线切线方向那样具有明确的几何特征。如果把结点看做是钉子，而把 Asymptote 的曲线看做是用若干钉子固定起来的一根弹簧或是硬橡皮绳，或许有助于理解所谓张力的概念。在不同的钉子之间，弹簧的张力越大，弹簧就越趋于直线的紧绷状态；反之，张力小时，弹簧就会因为松弛而自动张成圆弧状。张力用 **tension 张力值** 来设定，张力值是不小于 0.75 的实数，默认值 1 使得曲线最接近圆弧形状。例如：

```
draw(z4 {up} .. tension 0.75 .. {down} z6, heavyblue+2mm);
draw(z4 {up} .. tension 1 .. {down} z6, gray+2mm);
draw(z4 {up} .. tension 1.5 .. {down} z6, heavygreen+2mm);
```



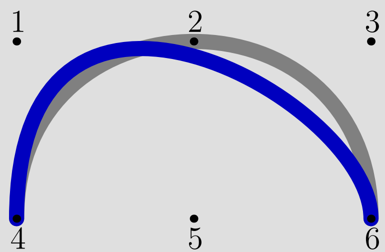
注意张力不仅影响两个结点之间的曲线形状，临近结点间的曲线形状也会受到影响。如：

```
draw(z4 .. tension 0.75 .. z2 .. z6, heavyblue+2mm);
draw(z4 .. tension 1 .. z2 .. z6, gray+2mm);
draw(z4 .. tension 1.5 .. z2 .. z6, heavygreen+2mm);
```



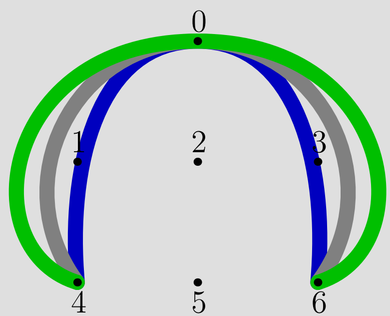
比弹簧或橡皮绳更强的是，Asymptote 曲线在结点两边的张力事实上也可以使用 **tension 起 and 末** 来分开设定：

```
draw(z4 {up} .. tension 1 .. {down} z6, gray+2mm);
draw(z4 {up} .. tension 0.75 and 2 .. {down} z6, heavyblue+2mm);
```



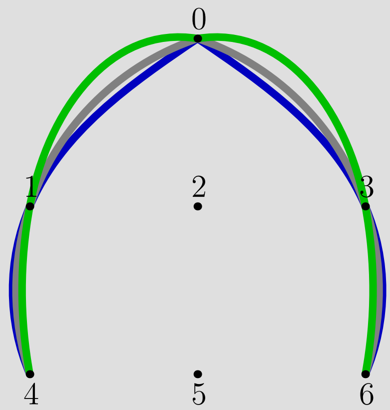
最后来看卷曲度。卷曲度影响曲线端点的弯曲程度，它的语法和设定曲线方向类似：**{curl 数值}**。卷曲度是一个非负值，其数值越大则曲线在端点的曲率越大，数值为 0 时则曲率趋于 0。在曲线的端点处，有默认的卷曲度 1，使得曲线接近圆弧。例如：

```
draw(z4 {curl 0} .. z0 .. {curl 0} z6, heavyblue+2mm);
draw(z4 {curl 1} .. z0 .. {curl 1} z6, gray+2mm);
draw(z4 {curl 5} .. z0 .. {curl 5} z6, heavygreen+2mm);
```



必须注意的是，卷曲度影响的是曲线的首尾端点的弯曲情况；而如果某个结点原本不是曲线的端点，那么曲线就会在此处折成尖角——特别地，如果相临端点都设置了卷曲度值，那么这两个结点之间就会是一段直线。例如：

```
draw(z4 .. z1 .. z0{curl 0} .. z3 .. z6, heavyblue+1mm);
draw(z4 .. z1 .. z0{curl 1} .. z3 .. z6, gray+1mm);
draw(z4 .. z1 .. z0{curl 5} .. z3 .. z6, heavygreen+1mm);
```

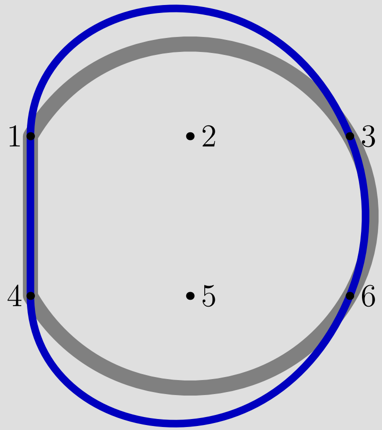


除了直线连接的符号 `--` 和曲线连接的符号 `..`，还有几个连接点坐标以得到曲线的连接符，它们分别是特殊的直线连接符 `---`、特殊的曲线连接符 `::` 以及将曲线首尾相接的 `&`。再加上连接几条不同曲线的 `^^`，就是全部连接曲线的符号了。

连接符 `---` 与 `--` 类似，都得到直线，单独使用 `---` 画出的直线和折线与使用 `--` 得到的结果并不能看出区别。但事实上，用 `---` 得到的并不是严格的直线折线，而只是非常接近直线的光滑曲线[2]。因而，如果把 `---` 与 `..` 配合使用，就能使得到的直线与曲线光滑地连接，而不像使用 `--` 那样得到不需要的尖角。例如：

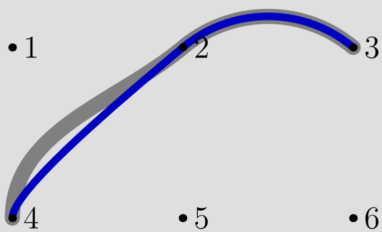
```
draw(z6 .. z4 -- z1 .. z3 .. cycle, gray+2mm);
draw(z6 .. z4 --- z1 .. z3 .. cycle, heavyblue+1mm);
```

[2]在 Asymptote 内部，连接符 `---` 是 `.. tension atleast infinity ..` 的缩写。**atleast** 关键字一般并不直接使用，故本文不作讨论，可参考 [2]。



连接符 :: 则与 .. 类似，都得到曲线[3]。不过连接符 :: 有一个特性，就是它将尽可能使两个坐标之间的曲线没有拐点（即尽可能避免两坐标间出现 S 形曲线，效果是在 .. 的基础上自动增加张力的结果）。例如：

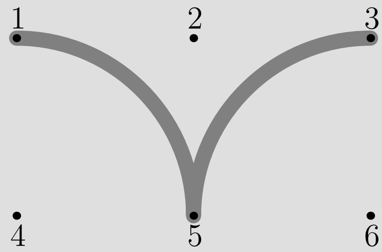
```
draw(z4{up} .. z2{dir(40)} .. z3, gray+2mm);
draw(z4{up} :: z2{dir(40)} .. z3, heavyblue+1mm);
```



& 连接符用于把几段首尾点依次重合的曲线直接相连，而不改变原来连接前曲线的形状。这事实上与使用重复点直接用 .. 生成尖角的效果相同，但在 Asymptote 中，连接符 & 会在生成的曲线中去除第一条曲线的尾结点（它应该是与第二条曲线的首结点重合的），因而不会出现像之前直接使用 .. 生成的曲线那样会有重复的结点。例如：

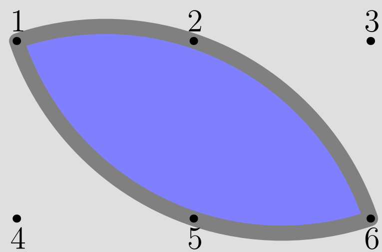
[3]在 Asymptote 内部，连接符 :: 是 .. **tension atleast** 1 .. 的缩写。Asymptote 的曲线功能和语法大多源自 METAPOST，不过在 METAPOST 中与 :: 功能相同的连接符是 ...。

```
draw(z1{right} .. {down} z5 & z5 {up} .. {right}z3, gray+2mm);
```



另一方面，相比直接单独绘制几条不相关的曲线，使用 & 连接符的一个重要作用是可以由此生成合适的闭曲线。例如：

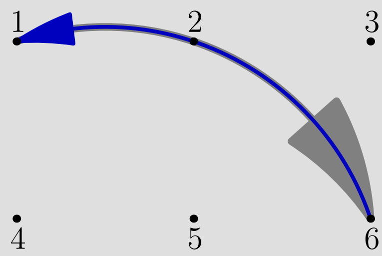
```
filldraw(z1 .. z2 .. z6 & z6 .. z5 .. z1 & cycle, lightblue, gray+2mm);
```



连接符 ^^ 在前面 section 1.2 中已经介绍过了。这里则指出它的一个特殊应用：在一个区域中“挖洞”填充。在数学中，平面上的一条简单闭曲线有的方向可以分为正负两种，逆时针方向为正，正时针方向为负。因此，在某个方向曲线围成的区域中用连接符 ^^ 加入一条反方向的闭曲线，就相当于在这个封闭区域中挖了一个洞。

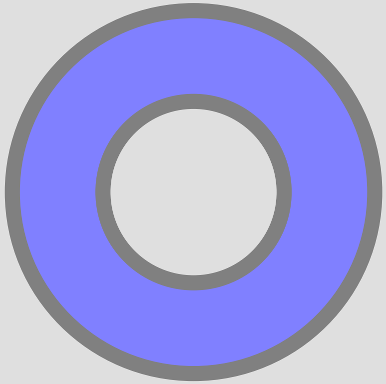
在给出相关的例子之前，我们首先给出 Asymptote 中把曲线方向逆转的函数 reverse(曲线)，它接受一条曲线（可以是路径或路向）作为参数，并返回此曲线的逆方向曲线。绘制时给曲线加上箭头可以明显看出其方向：

```
draw(z1 .. z2 .. z6, gray+1mm, Arrow);
draw(reverse(z1 .. z2 .. z6), heavyblue+0.5mm, Arrow);
```



于是，利用这种机制，在大圆内加上一个反方向的圆，把这两条曲线用 ^^ 连接，就可以用来填充得到环形：

```
size(5cm, 0);
filldraw(circle((0,0), 2) ^^ reverse(circle((0,0), 1)), lightblue, gray+2mm);
```

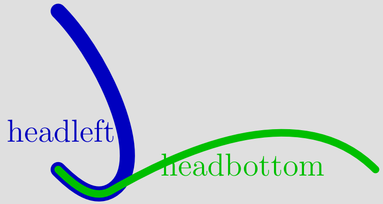


需要说明的是，使用 & 连接两条曲线，得到的结果是一条连续的曲线；而使用 ^^ 连接两条曲线，得到的结果并不是一条曲线，而只是把两条曲线合并在一起的曲线数组（数组在稍后的章节介绍）。因而它们在使用时是不同的。

上面介绍的内容，差不多就是 Asymptote 中全部基本的曲线控制方式了。有了这些， André 老师就可以按照 section 2.1 中的方式，精确地设计羊头的轮廓了。他分别定义羊头左边和下面的两根线：

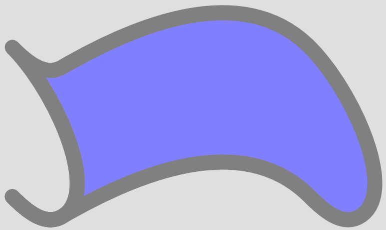
```
guide headleft = (0,1.5){SE} .. tension 1.4 .. (0.5,-0.2){dir(-150)} ..
      {NW}(0,0);
```

```
guide headbottom = (0,0){SE} .. {dir(30)}(0.5,-0.2) ..{SE}(3,0);
```



这里的两条曲线虽然看起来定义比较复杂，但它们都只是由三个结点连接而成的，定义更为直观，且在数学上更加简单。此外，明确限定了两条曲线在结点处的切线方向，可以保证它们可以精确拼接。于是，通过平移变换、使用 `reverse` 函数逆向并使用连接符 & 的拼接，就可以得到完整的羊头轮廓了：

```
guide head = headleft & headbottom &
    shift(3,0)*reverse(headleft) & shift(0,1.5)*reverse(headbottom) & cycle;
```

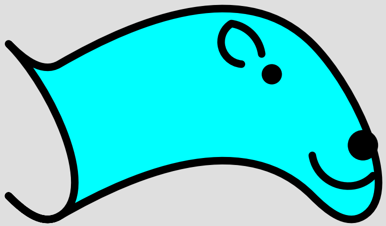


下面，羊的眼睛（eye）、耳朵（ear）、鼻子（muzzle）和嘴（mouth）就比较容易得到了：

```
guide eye = circle((2.6,1.2), 0.1);
guide ear = (2.3,1.3) .. (2.1,1.5) .. (2.2,1.7)
    & (2.2,1.7) .. (2.4,1.6) .. (2.5,1.4);
guide muzzle = circle((3.5,0.5), 0.15);
guide mouth = (3,0.4) .. (3.4,0.1) .. (3.6,0.2);
```

于是，André 就这样画出了一个完整的青色羊头：

```
size(5cm, 0);
filldraw(head, cyan, linewidth(1mm));
fill(eye ^^ muzzle);
draw(ear ^^ mouth, linewidth(1mm));
```



挑剔的 André 对这个羊头的曲线感到满意，因为作为一个铺砌图形，它绘制得完全精确。一个羊头已经可以很好地画出来了，剩下的问题就是重复地把这个羊头铺满平面，开始“铺砖”了。

2.4 循环与条件判断

要在平面上铺满相同的图形，就需要反复地在不同的位置进行绘制。Asymptote 中可以使用循环语句来完成这种重复的工作。

最常用的循环语句是 **for** 循环，其基本语法是：

```
for (初始化; 执行条件; 循环增量)
    循环语句体
```

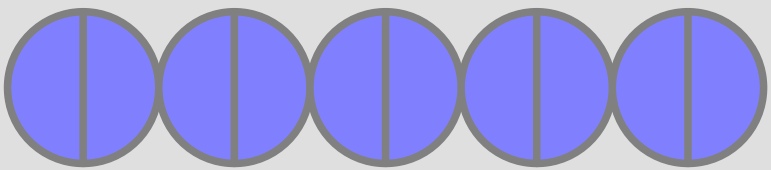
在一个循环语句中，Asymptote 首先运行初始化语句，随后开始不断运行循环语句体，每次运行语句体前检查执行条件以判断是否开始，而在运行语句体后运行循环增量的语句。例如下面的语句会画出起始角度为 0°, 30°, 60°, 90° 的四条曲线：

```
for (real angle = 0; angle <= 90; angle += 30)
    draw( (0,0){dir(angle)} .. {right}(6cm,0), gray+1mm );
```

在初始化部分定义了一个实数（**real**）类型的变量 angle，初始值为 0；进入循环的执行条件是角度变量 angle 不大于直角；而在每次循环体画完曲线后，变量 angle 的值就增加 30°。这里复合赋值运算符 += 表示在变量原有的基础上累加， var += x 就等价于 var = var + x，类似地减法、乘除法、取余数和乘方也有 -=、*=、 /=、%=、^= 这几种对应的赋值形式。对于整数 （**int**）类型的变量，还可以使用 ++var 和 --var 分别表示 var += 1 和 var -= 1。

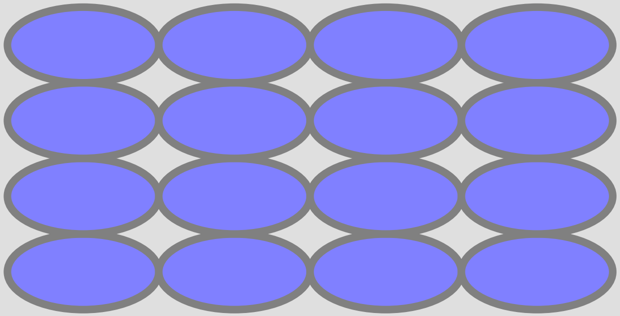
如果循环体的语句不止一个，可以用花括号把这些语句括起来。如：

```
for (int i = 0; i < 5; ++i) {
    filldraw(circle((2i*cm,0), 1cm), lightblue, gray+1mm);
    draw( (2i*cm,-1cm) -- (2i*cm,1cm), gray+1mm);
}
```



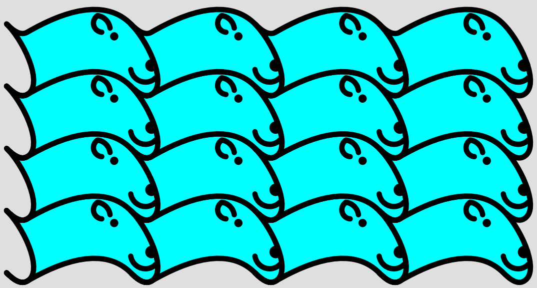
平面铺砌图形需要在横向和纵向两个方向重复延伸，因而可以使用两层嵌套的循环语句来完成这种图形：

```
for (real x = 0; x < 8cm; x += 2cm)
    for (real y = 0; y < 4cm; y += 1cm)
        filldraw(ellipse((x,y), 1cm, 0.5cm), lightblue, gray+1mm);
```



因此，有了前面的准备工作，André 就很容易通过一个双重循环来做出羊头的铺砌形状来：

```
// ……相关曲线的定义同前
size(7cm, 0);
for (int x = 0; x < 4; ++x) {
    for (int y = 0; y < 4; ++y) {
        transform pos = shift(3x, 1.5y);
        filldraw( pos * head, cyan, linewidth(2bp));
        fill( pos * (eye ^^ muzzle) );
        draw( pos * (ear ^^ mouth), linewidth(2bp));
    }
}
```



在这里，André 使用关于整数变量 x 和 y 的二重循环来控制整个铺砌图。在循环体中使用了变换类型 **transform** 的变量 pos 来保存从原点到每一块铺砌图形元的平移变换。在绘图时，给每条曲线都使用 pos 进行变换，就可以在正确的位置绘图，而不用直接修改原来的曲线。

现在，André 几乎已经得到了和图 2.1 相同的图形了。不过在 图 2.1 中，羊头的颜色是随位置交错变化的。一个笨办法是分别用两组循环来画出黄色和青色的羊头，但这会带来很多重复的代码。因此 André 决定在循环体内做一点手脚：出于数学教师的敏感，他发现羊头颜色的规律可以由整数 $x+y$ 的奇偶性来判定，于是只要在循环体内增加一点判断……

Asymptote 的条件判断语句正好可以完成这项工作。条件判断语句的基本语法是：

```
// 只有一个分支
if (条件)
    条件为真执行的语句块
// 有 else 块，有两个分支
if (条件)
    条件为真执行的语句块
else
    条件为假执行的语句块
```

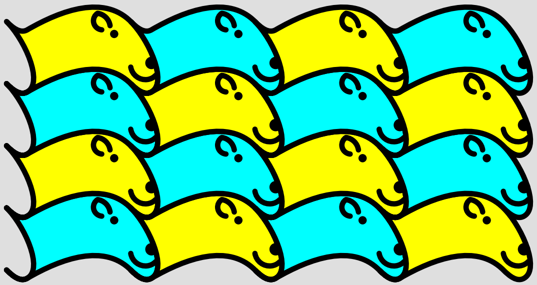
其中，可以用花括号括起多行的语句块。因而，可以使用如下的语句来通过判断 $x+y$ 的奇偶性来决定画笔的颜色：

```
pen color;
if ( (x+y) % 2 == 0 )   // 判断 x+y 是否为偶数
    color = cyan;
else
    color = yellow;
```

把上面的控制颜色的判断语句用到前面的循环体中，就可以得到交错色彩的铺砌图了：

```
// ……相关曲线的定义同前
size(7cm, 0);
for (int x = 0; x < 4; ++x) {
    for (int y = 0; y < 4; ++y) {
```

```
        transform pos = shift(3x, 1.5y);
        pen color;
        if ( (x+y) % 2 == 0 )
            color = cyan;
        else
            color = yellow;
        filldraw( pos * head, color, linewidth(2bp));
        fill( pos * (eye ^^ muzzle) );
        draw( pos * (ear ^^ mouth), linewidth(2bp));
    }
}
```



至此，André 大松一口气，他的铺砌图形基本上就绘制完成了。不过在他结束这一切的时候，他的同事，Asymptote 的专家 Philippe 告诉他，其实有关条件判断的语句还可以使用一个条件表达式来简化。在这里，要判断的条件比较简单，而判断的结果也只是青、黄两种颜色的值，因此，可以使用条件运算符 ? : 来代替。条件运算符是一个特殊的三元运算符，它的语法是：

```
条件 ? 条件为真的值 : 条件为假的值
```

因此前面的颜色判断和赋值语句就可以简化为：

```
pen color = (i+j)%2==0 ? cyan : yellow;
```

现在，André 的铺砌图形就完成了。下面只要把他的图形剪裁一下，就大工告成了。

2.5 图形的剪裁

André 最后的问题是把整个铺砌图形剪裁为椭圆形。为此，我们就需要引入在画线（draw）、填充（fill）、标注（label）之外的最后一种基本绘图命令，即剪裁（clip）。剪裁命令的基本语法是：

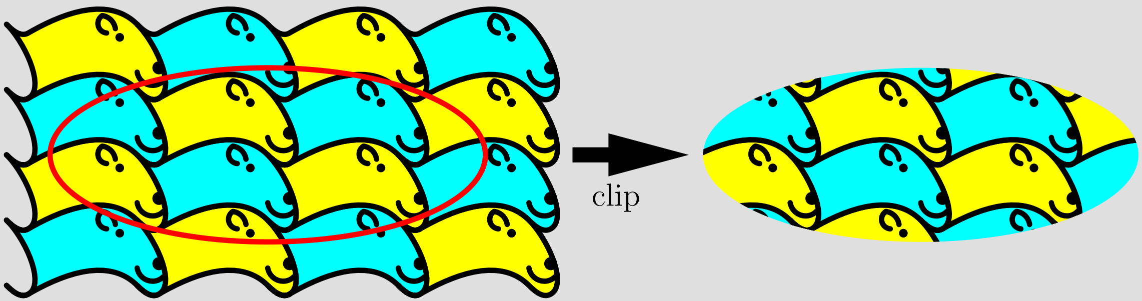
```
clip(图, 曲线);     // 图的默认值是当前图 currentpicture
```

如果省略 **picture** 类型的图参数，则剪裁命令会把当前正在绘制的图用给定的曲线裁剪开来；显示指定图参数则只剪裁给定的图中的内容。

剪裁命令的效果正是 André 需要的，为了用椭圆形剪裁图形，André 只需要增加一句

```
clip(ellipse((6,3), 5, 2));
```

就得到下图右侧的剪裁效果：



最后，André 得到了完整画出图 2.1 的全部代码：

```
// André 的铺砌图案

// 全局图形缩放
size(15cm, 0);

// 定义羊头曲线
```

```
guide headleft = (0,1.5){SE} .. tension 1.4 .. (0.5,-0.2){dir(-150)} ..
     {NW}(0,0);
guide headbottom = (0,0){SE} .. {dir(30)}(0.5,-0.2) ..{SE}(3,0);
guide head = headleft & headbottom &
    shift(3,0)*reverse(headleft) & shift(0,1.5)*reverse(headbottom) & cycle;
guide eye = circle((2.6,1.2), 0.1);
guide ear = (2.3,1.3) .. (2.1,1.5) .. (2.2,1.7)
    & (2.2,1.7) .. (2.4,1.6) .. (2.5,1.4);
guide muzzle = circle((3.5,0.5), 0.15);
guide mouth = (3,0.4) .. (3.4,0.1) .. (3.6,0.2);

// 绘制铺砌图形
for (int x = 0; x < 8; ++x) {
    for (int y = 0; y < 8; ++y) {
        transform pos = shift(3x, 1.5y);
        pen color;
        if ( (x+y) % 2 == 0 )
            color = cyan;
        else
            color = yellow;
        filldraw(pos * head, color, linewidth(2bp));
        fill(pos * (eye ^^ muzzle) );
        draw(pos * (ear ^^ mouth), linewidth(2bp));
    }
}

// 剪裁为椭圆形
clip(ellipse((12,6), 10, 5));
```

附录 A　GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

A.1　Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

A.2 Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

(a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

(b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

(c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

(d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law.

If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

    A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

A.3 End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.

索引

参考文献

[1] Andy Hammerlindl, John Bowman, and Tom Prince. *Asymptote: the Vector Graphics Language*, 1.88 edition, 2009. 这是 Asymptote 的官方手册。刘海洋给出了文档的一个翻译，见 `http://code.google.com/p/asy4cn/`。.

[2] Donald Ervin Knuth. *The METAFONTbook*, volume C of *Computers & Typesetting*. Addison Wesley, 1986.

[3] Timothy Van Zandt. *PSTricks: PostScript macros for Generic TEX*, 97 edition, July 2003.

[4] 曲安京. 商高、趙爽與劉徽關於勾股定理的證明. 數學傳播, 20(3), 1998.