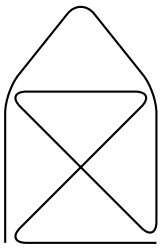
Part I
教程和指导

by Till Tantau

为了帮你入门 Ti*k*Z，本手册没有立刻给出长长的安装和配置过程，而是直接从教程开始。这些教程解释了该系统所有基本特性和部分高级特性，并不深入所有细节。这部分还指导你在用 Ti*k*Z 绘图时，如何继续前进。



```
\tikz \draw[thick,rounded corners=8pt]
  (0,0) -- (0,2) -- (1,3.25) -- (2,2) -- (2,0) -- (0,2) -- (2,2) -- (0,0) -- (2,0);
```

1 Tutorial: Euclid's Amber Version of the *Elements*

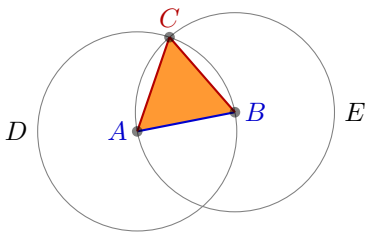
在这第三篇教程中，我们看看如何用 Ti*k*Z 实现几何作图。

Euclid 最近在忙着写一套新书，书名暂定为“几何原本”（Euclid 不是很确定这个书名是否能把主旨传达给后人，但是他打算在交付给出版商之前修改一下书名）。到目前为止，他用莎草纸写字画图，但是他的出版商突然要求他必须提交电子版。Euclid 跟出版商争辩道，电子产品要在几千年后才会发明出来，然而出版商告诉他，莎草纸不再是尖端技术了，他必须得跟上现代工具的步伐。

Euclid 略带不满，准备把他在莎草纸写的东西转到琥珀上，该部分题为“第 I 卷，命题 1”。

1.1 卷 I，命题 1

他在莎草纸上的图是这样的：[1] [2] [3]



命题 1
在一条已知有限直线上作一个等边三角形.

设 AB 是已知有限直线. 要求在直线 AB 上作一个等边三角形.
以 A 为中心，AB 为半径作圆 BCD；再以 B 为中心，BA 为半径作圆 ACE；由两圆交点 C 分别连直线到 A、B，得 CA、CB.
因为点 A 为圆 CDB 的圆心，所以 AC 等于 AB. 又因为 B 为圆 CAE 的圆心，所以 BC 等于 BA. 然而已经证明了 AC 等于 AB；所以线段 AC 、BC 都等于 AB. 而且等于同量的量彼此相等，因此三条线段 AC、AB和 BC 彼此相等. 所以三角形 ABC 是等边的，并且已经在给定的有限直线 AB 上作出了该三角形.

让我们看看，Euclid 怎么把这张图变成 Ti*k*Z 代码。

1.1.1 设置环境

如之前教程所述，Euclid 需要载入 Ti*k*Z 以及一些库，这些库包括 `calc`、`intersections`、`through` 和 `backgrounds`。根据格式不同，Euclid 要在序言区加上下列语句之一：

```
% 对于 LaTeX:
\usepackage{tikz}
\usetikzlibrary{calc,intersections,through,backgrounds}
```

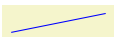
```
% 对于 plain TeX:
\input tikz.tex
\usetikzlibrary{calc,intersections,through,backgrounds}
```

```
% 对于 ConTeXt:
\usemodule[tikz]
\usetikzlibrary[calc,intersections,through,backgrounds]
```

1.1.2 线 *AB*

图中 Euclid 想最先画的部分是线 AB。很简单，`\draw (0,0) -- (2,1);` 就能搞定。但是，Euclid 不想在 A 和 B 后面加上 $(0,0)$ 和 $(2,1)$，他想只写上 `A` 和 `B`。事实上，他的书的主旨是，点 A 和 B 可以是任意的，其他的点（比如 C）可以根据它们的位置构造出来，Euclid 不需要直接写出 C 的坐标。

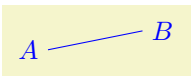
所以 Euclid 用 `\coordinate` 命令定义两个坐标：



```
\begin{tikzpicture}
  \coordinate (A) at (0,0);
  \coordinate (B) at (1.25,0.25);

  \draw[blue] (A) -- (B);
\end{tikzpicture}
```

[1] 该段文本摘自 David E. Joyce 漂亮的交互式欧几里得几何原本，可以在他 Clark 大学的网站上找到。
[2] 该段文本原文链接：https://mathcs.clarku.edu/~djoyce/java/elements/bookI/propI1.html
[3] 该段译文部分参考了：欧几里得 著，兰纪正、朱恩宽 译. 欧几里得 • 几何原本. 陕西科学技术出版社, 2003. ISBN 9787536903579.

非常简单。这里少了坐标的标签，Euclid 不想标在点上面，而是在点旁边。他决定用 label 选项：




```
\begin{tikzpicture}
  \coordinate [label=left:\textcolor{blue}{$A$}]  (A) at (0,0);
  \coordinate [label=right:\textcolor{blue}{$B$}] (B) at (1.25,0.25);

  \draw[blue] (A) -- (B);
\end{tikzpicture}
```

Euclid 这时在想，要是点 A 和 B 可以“随机”一点就更好了。关于这些点的位置，Euclid 和读者都不能犯下“想当然”的错误。Euclid 很高兴地了解到，Ti*k*Z 里有一个 rand 函数可以满足要求：它能生成一个 -1 到 1 之间的数。既然 Ti*k*Z 能做些数学运算，Euclid 就可以把点坐标写成下面这样：

```
\coordinate [...] (A) at (0+0.1*rand,0+0.1*rand);
\coordinate [...] (B) at (1.25+0.1*rand,0.25+0.1*rand);
```

这个效果不错，但是 Euclid 并不是很满意，因为他想让“主坐标” $(0,0)$ 和 $(1.25,0.25)$ 同扰动 $0.1(rand, rand)$ “分离开来”。也就是说，他希望将坐标 A 指定成这样，“点 $(0,0)$ 加上向量 $(rand, rand)$ 的十分之一”。

实际上，calc 库来能让他做这类运算。载入该库后，你可以用特殊的坐标，始于 ($ 止于 $)，而不只是 (和)。你可以对这些特殊的坐标进行线性组合。（注意到 $ 符号只是表示开始“运算”，而不是输入数学公式。）

关于坐标的新代码如下：

```
\coordinate [...] (A) at ($ (0,0) + .1*(rand,rand) $);
\coordinate [...] (B) at ($ (1.25,0.25) + .1*(rand,rand) $);
```

注意，如果在这类计算中，一个坐标包含小数（比如 .1），那么你必须在坐标的左圆括号之前加上 *。这些运算可以嵌套。

1.1.3 围绕点 *A* 的圆

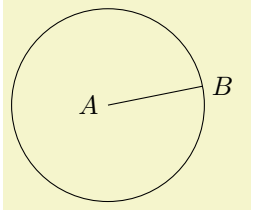
第一个棘手的构造是围绕点 A 的圆。我们之后会介绍一个非常简单的方法，但是首先让我们用“困难的”方式实现它。

思路如下：我们画一个围绕点 A 的圆，其半径由线段 AB 的长度决定。困难在于计算这条线段的长度。

可以通过两个点子解决这个问题：第一个，我们可以用 ($ (A) - (B) $) 表示 A 和 B 相差的向量。我们需要的是这个向量的长度。第二个，给定两个数 x 和 y，我们可以在数学表达式中写 veclen(x,y)，这会返回数值 $\sqrt{x^2+y^2}$，也就是想要的长度。

剩下的唯一问题就是如何获得向量 AB 的 x 轴和 y 轴坐标。为此我们需要引入一个新概念：let 操作。let 操作可以在路径中这样一些地方插入，比如直线指向的位置或者移动的目标位置。let 操作的效果是计算一些坐标并将其结果赋给特定的宏，这些宏方便了对坐标的 x 和 y 值的访问。

Euclid 可以这样写：




```
\begin{tikzpicture}
  \coordinate [label=left:$A$]  (A) at (0,0);
  \coordinate [label=right:$B$] (B) at (1.25,0.25);
  \draw (A) -- (B);

  \draw (A) let
              \p1 = ($ (B) - (A) $)
            in
              circle ({veclen(\x1,\y1)});
\end{tikzpicture}
```

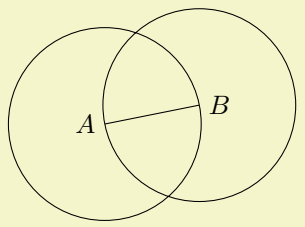
let 操作的每一次赋值都从 \p 开始，通常跟着一个 ⟨数字⟩，接着是一个等号和一个坐标。计算好坐标后，结果保存在内部，之后你就可以用下面的表达式：

1. \x⟨数字⟩ 得到对应点的 x 坐标。
2. \y⟨数字⟩ 得到对应点的 y 坐标。
3. \p⟨数字⟩ 等同于 \x⟨数字⟩,\y⟨数字⟩。

你可以在 let 操作中赋很多值，只要用逗号隔开。后面的赋值可以使用前面的赋值结果。

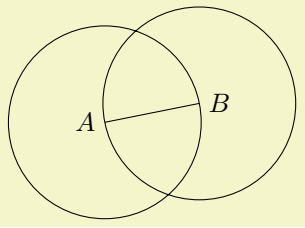
注意，\p1 并不是通常意义上的坐标，它只是扩展成 10pt,20pt 这样的字符串。因此，你不能写 (\p1.center) 这样的语句，因为这个只会扩展成 (10pt,20pt.center)，这条语句没有意义。

下一步，我们想同时将两个圆画出来。两个圆半径都是 veclen(\x1,\y1)，因此自然只需要计算一次。这里我们还可以用一个 let 操作：不是写 \p1 = ...，而是写 \n2 = ...。这里的 n 表示“数”（number），同理，p 表示“点”（point）。给数赋值需要在外面加上花括号。




```
\begin{tikzpicture}
  \coordinate [label=left:$A$]  (A) at (0,0);
  \coordinate [label=right:$B$] (B) at (1.25,0.25);
  \draw (A) -- (B);

  \draw let \p1 = ($ (B) - (A) $),
            \n2 = {veclen(\x1,\y1)}
        in
          (A) circle (\n2)
          (B) circle (\n2);
\end{tikzpicture}
```

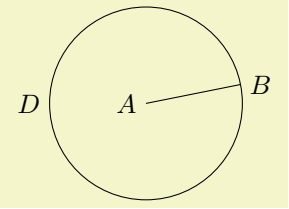
看了上例，你可能会想，如果写 \n1 会得到什么？答案是会显示它没有定义。宏 \p、\x 和 \y 引用逻辑上的同一个点，而宏 \n 则有“自己的命名空间”。我们甚至可以将例中的 \n2 替换成 \n1，效果一样。事实上，跟在这些宏后面的数字只是普通的 TEX 参数。我们还可以用更长的名字，不过需要加上花括号：




```
\begin{tikzpicture}
  \coordinate [label=left:$A$]  (A) at (0,0);
  \coordinate [label=right:$B$] (B) at (1.25,0.25);
  \draw (A) -- (B);

  \draw let \p1        = ($ (B) - (A) $),
            \n{radius} = {veclen(\x1,\y1)}
        in
          (A) circle (\n{radius})
          (B) circle (\n{radius});
\end{tikzpicture}
```

在本节开始我们答应过，有个更简单的方法画出想要的圆。这个技巧就是用 through 库。顾名思义，它的作用就是通过一个给定的点创建形状。

我们要找的选项是 circle through。这个选项传给一个节点，产生如下效果：首先，它将节点的内外间距置零；然后，将节点的形状设为圆 circle；最后，它根据传给 circle through 的参数决定圆的半径，本质上，计算该半径的方法和上面一样。

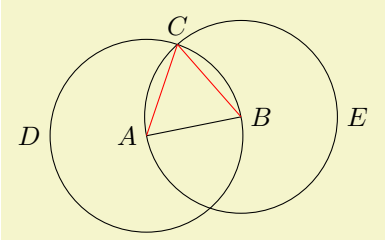



```
\begin{tikzpicture}
  \coordinate [label=left:$A$]  (A) at (0,0);
  \coordinate [label=right:$B$] (B) at (1.25,0.25);
  \draw (A) -- (B);

  \node [draw,circle through=(B),label=left:$D$] at (A) {};
\end{tikzpicture}
```

1.1.4 圆的交点

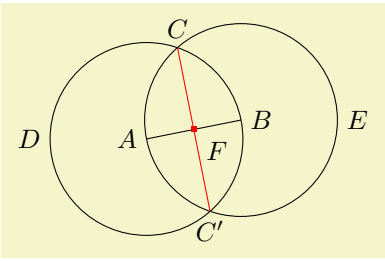
Euclid 现在可以画线和圆了。最后一个问题就是计算两圆的交点，这就涉及到你是否想“手算”它。好在 intersections 库可以让我们计算任意路径的交点。

思路很简单：首先用 name path 选项“命名”两条路径；接着，你可以在之后用 name intersections 选项，这会在路径之间的所有交点处创建坐标，名为 intersection-1、intersection-2 等等。Euclid 将两圆交点命名为 D 和 E（恰好和节点重名，不过节点和路径拥有不同的“命名空间”）。




```
\begin{tikzpicture}
  \coordinate [label=left:$A$]  (A) at (0,0);
  \coordinate [label=right:$B$] (B) at (1.25,0.25);
  \draw (A) -- (B);

  \node (D) [name path=D,draw,circle through=(B),label=left:$D$]  at (A) {};
  \node (E) [name path=E,draw,circle through=(A),label=right:$E$] at (B) {};

  % 给坐标命名，但是什么都不画：
  \path [name intersections={of=D and E}];

  \coordinate [label=above:$C$] (C) at (intersection-1);

  \draw [red] (A) -- (C);
  \draw [red] (B) -- (C);
\end{tikzpicture}
```

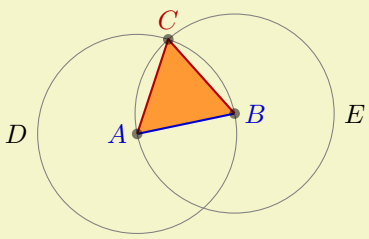
这可以进一步简写：name intersections 有一个可选参数 by，你可以用它指定坐标的名称和选项，这让代码更紧凑。虽然 Euclid 画现在这张图还用不到，但是画线段 AB 的平分线时，用这个参数只需要一小步。




```
\begin{tikzpicture}
  \coordinate [label=left:$A$]  (A) at (0,0);
  \coordinate [label=right:$B$] (B) at (1.25,0.25);
  \draw [name path=A--B] (A) -- (B);

  \node (D) [name path=D,draw,circle through=(B),label=left:$D$]  at (A) {};
  \node (E) [name path=E,draw,circle through=(A),label=right:$E$] at (B) {};

  \path [name intersections={of=D and E, by={[label=above:$C$]C, [label=below:$C'$]C'}}];

  \draw [name path=C--C',red] (C) -- (C');

  \path [name intersections={of=A--B and C--C',by=F}];
  \node [fill=red,inner sep=1pt,label=-45:$F$] at (F) {};
\end{tikzpicture}
```

1.1.5 完整代码

回到 Euclid 的代码，人生苦短，他写了几个宏。比如用 \A 表示一个蓝色的 A，他还用了 background 层，在结尾处画了一个三角形，并置于最底层。



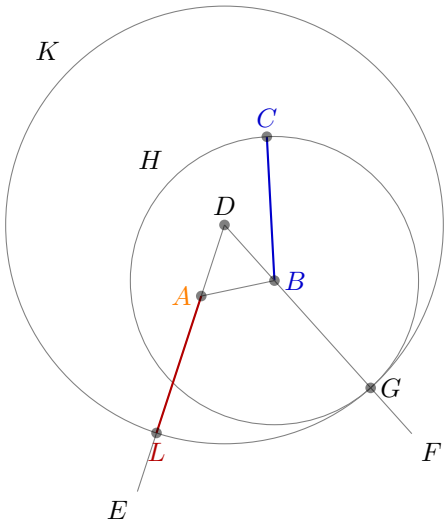
命题 1
在一条已知有限直线上作一个等边三角形.

设 AB 是已知有限直线. ...

```
\begin{tikzpicture}[thick,help lines/.style={thin,draw=black!50}]
  \def\A{\textcolor{input}{$A$}}     \def\B{\textcolor{input}{$B$}}
  \def\C{\textcolor{output}{$C$}}    \def\D{$D$}
  \def\E{$E$}

  \colorlet{input}{blue!80!black}    \colorlet{output}{red!70!black}
  \colorlet{triangle}{orange}

  \coordinate [label=left:\A]  (A) at ($ (0,0) + .1*(rand,rand) $);
  \coordinate [label=right:\B] (B) at ($ (1.25,0.25) + .1*(rand,rand) $);

  \draw [input] (A) -- (B);

  \node [name path=D,help lines,draw,label=left:\D]   (D) at (A) [circle through=(B)] {};
  \node [name path=E,help lines,draw,label=right:\E]  (E) at (B) [circle through=(A)] {};

  \path [name intersections={of=D and E,by={[label=above:\C]C}}];

  \draw [output] (A) -- (C) -- (B);

  \foreach \point in {A,B,C}
    \fill [black,opacity=.5] (\point) circle (2pt);

  \begin{pgfonlayer}{background}
    \fill[triangle!80] (A) -- (C) -- (B) -- cycle;
  \end{pgfonlayer}

  \node [below right, text width=10cm,align=justify] at (4,3) {
    \small
    \textbf{命题 1}\par
    \emph{在一条已知\textcolor{input}{有限直线}上作一个\textcolor{triangle}{等边三角形}.}
    \par\vskip1em
    设 \A\B\ 是已知\textcolor{input}{有限直线}. \dots
  };
\end{tikzpicture}
```

1.2 卷 I，命题 2

《几何原本》中的第二个命题如下：



命题 2
以一个已知点作为端点，作一条线段与已知线段相等.

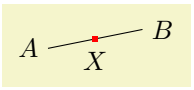
设 A 为已知点，BC 为已知线段. 要求以 A 为端点，作一条线段与已知线段 BC 相等.
连接点 A 和 B 得线段 AB，并在其上作一等边三角形 DAB.
延长 DA 和 DB 为直线 AE 和 BF。以 B 为中心，BC 为半径，作圆 CGH，再以 D 为中心，DG 为半径，作圆 GKL.
因为点 B 是圆 CGH 的圆心，所以 BC 等于 BG. 同样，因为点 D 是圆 GKL 的圆心，所以 DL 等于 DG. 又因为 DA 等于 DB，因此线段之差 AL 等于线段之差 BG. 然而已经证明了 BC 等于 BG，所以 AL 和 BC 均等于 BG. 而且等于同量的量彼此相等，所以 AL 也等于 BC.
所以，由已知点 A 作出了线段 AL ，与已知线段 BC 相等.

1.2.1 Using Partway Calculations for the Construction of D

Euclid 在构造点 D 时，“参考”了命题 1 中的过程。现在，我们也可以简单重复这些构造过程，但是似乎有点麻烦，因为我们得画出所有这些圆，完成所有复杂的构造。

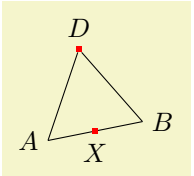
因此，TikZ 提供了简便方法。首先，有一个简单的语法，可以计算从 p 到 q 连线上的一点：将这两个点放到坐标计算环境中——别忘了首尾的 ($ 和 $)——然后用 !⟨分比⟩! 将它们组合起来。⟨分比⟩ 为 0 表示前面的坐标，⟨分比⟩为 1 表示后面的坐标，0 到 1 之间的值表示两点连线上的一点。所以，这个语法类似于 xcolor 中混合颜色的语法。

这里展示了如何计算线段 AB 的中点：



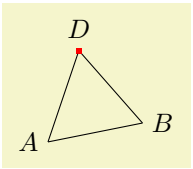

```
\begin{tikzpicture}
  \coordinate [label=left:$A$]  (A) at (0,0);
  \coordinate [label=right:$B$] (B) at (1.25,0.25);
  \draw (A) -- (B);
  \node [fill=red,inner sep=1pt,label=below:$X$] (X) at ($ (A)!.5!(B) $) {};
\end{tikzpicture}
```

The computation of the point D in Euclid's second proposition is a bit more complicated. It can be expressed as follows: Consider the line from X to B. Suppose we rotate this line around X for 90° and then stretch it by a factor of $\sin(60°) \cdot 2$. This yields the desired point D. We can do the stretching using the partway modifier above, for the rotation we need a new modifier: the rotation modifier. The idea is that the second coordinate in a partway computation can be prefixed by an angle. Then the partway point is computed normally (as if no angle were given), but the resulting point is rotated by this angle around the first point.




```
\begin{tikzpicture}
  \coordinate [label=left:$A$]  (A) at (0,0);
  \coordinate [label=right:$B$] (B) at (1.25,0.25);
  \draw (A) -- (B);
  \node [fill=red,inner sep=1pt,label=below:$X$] (X) at ($ (A)!.5!(B) $) {};
  \node [fill=red,inner sep=1pt,label=above:$D$] (D) at
    ($ (X) ! {sin(60)*2} ! 90:(B) $) {};
  \draw (A) -- (D) -- (B);
\end{tikzpicture}
```

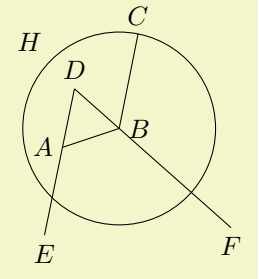
Finally, it is not necessary to explicitly name the point X. Rather, again like in the xcolor package, it is possible to chain partway modifiers:




```
\begin{tikzpicture}
  \coordinate [label=left:$A$]  (A) at (0,0);
  \coordinate [label=right:$B$] (B) at (1.25,0.25);
  \draw (A) -- (B);
  \node [fill=red,inner sep=1pt,label=above:$D$] (D) at
    ($ (A) ! .5 ! (B) ! {sin(60)*2} ! 90:(B) $) {};
  \draw (A) -- (D) -- (B);
\end{tikzpicture}
```

1.2.2 Intersecting a Line and a Circle

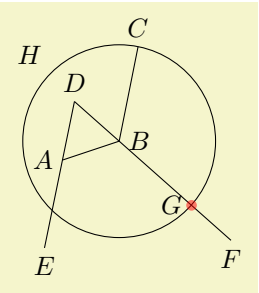
The next step in the construction is to draw a circle around B through C, which is easy enough to do using the `circle through` option. Extending the lines DA and DB can be done using partway calculations, but this time with a part value outside the range $[0, 1]$:




```
\begin{tikzpicture}
  \coordinate [label=left:$A$]  (A) at (0,0);
  \coordinate [label=right:$B$] (B) at (0.75,0.25);
  \coordinate [label=above:$C$] (C) at (1,1.5);
  \draw (A) -- (B) -- (C);
  \coordinate [label=above:$D$] (D) at
    ($ (A) ! .5 ! (B) ! {sin(60)*2} ! 90:(B) $) {};
  \node (H) [label=135:$H$,draw,circle through=(C)] at (B) {};
  \draw (D) -- ($ (D) ! 3.5 ! (B) $) coordinate [label=below:$F$] (F);
  \draw (D) -- ($ (D) ! 2.5 ! (A) $) coordinate [label=below:$E$] (E);
\end{tikzpicture}
```

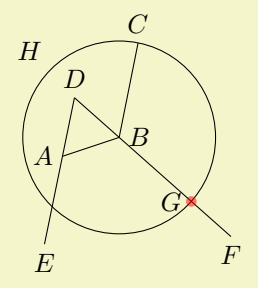
We now face the problem of finding the point G, which is the intersection of the line BF and the circle H. One way is to use yet another variant of the partway computation: Normally, a partway computation has the form $\langle p\rangle\,!\,\langle factor\rangle\,!\,\langle q\rangle$, resulting in the point $(1-\langle factor\rangle)\langle p\rangle + \langle factor\rangle\langle q\rangle$. Alternatively, instead of $\langle factor\rangle$ you can also use a $\langle dimension\rangle$ between the points. In this case, you get the point that is $\langle dimension\rangle$ away from $\langle p\rangle$ on the straight line to $\langle q\rangle$.

We know that the point G is on the way from B to F. The distance is given by the radius of the circle H. Here is the code for computing H:



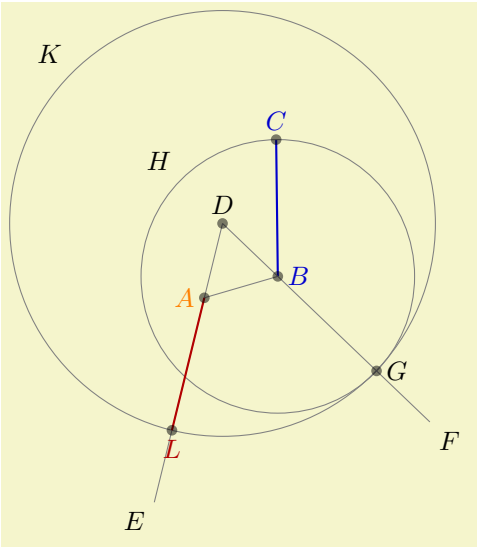

```
  \node (H) [label=135:$H$,draw,circle through=(C)] at (B) {};
  \path let \p1 = ($ (B) - (C) $) in
    coordinate [label=left:$G$] (G) at ($ (B) ! veclen(\x1,\y1) ! (F) $);
  \fill[red,opacity=.5] (G) circle (2pt);
```

However, there is a simpler way: We can simply name the path of the circle and of the line in question and then use `name intersections` to compute the intersections.




```
  \node (H) [name path=H,label=135:$H$,draw,circle through=(C)] at (B) {};
  \path [name path=B--F] (B) -- (F);
  \path [name intersections={of=H and B--F,by={[label=left:$G$]G}}];
  \fill[red,opacity=.5] (G) circle (2pt);
```

1.2.3 The Complete Code




```
\begin{tikzpicture}[thick,help lines/.style={thin,draw=black!50}]
  \def\A{\textcolor{orange}{$A$}}   \def\B{\textcolor{input}{$B$}}
  \def\C{\textcolor{input}{$C$}}    \def\D{$D$}
  \def\E{$E$}                       \def\F{$F$}
  \def\G{$G$}                       \def\H{$H$}
  \def\K{$K$}                       \def\L{\textcolor{output}{$L$}}

  \colorlet{input}{blue!80!black}    \colorlet{output}{red!70!black}

  \coordinate [label=left:\A]  (A) at ($ (0,0) + .1*(rand,rand) $);
  \coordinate [label=right:\B] (B) at ($ (1,0.2) + .1*(rand,rand) $);
  \coordinate [label=above:\C] (C) at ($ (1,2) + .1*(rand,rand) $);

  \draw [input] (B) -- (C);
  \draw [help lines] (A) -- (B);

  \coordinate [label=above:\D] (D) at ($ (A)!.5!(B) ! {sin(60)*2} ! 90:(B) $);

  \draw [help lines] (D) -- ($ (D)!3.75!(A) $) coordinate [label=-135:\E] (E);
  \draw [help lines] (D) -- ($ (D)!3.75!(B) $) coordinate [label=-45:\F] (F);

  \node (H) at (B) [name path=H,help lines,circle through=(C),draw,label=135:\H] {};
  \path [name path=B--F] (B) -- (F);
  \path [name intersections={of=H and B--F,by={[label=right:\G]G}}];

  \node (K) at (D) [name path=K,help lines,circle through=(G),draw,label=135:\K] {};
  \path [name path=A--E] (A) -- (E);
  \path [name intersections={of=K and A--E,by={[label=below:\L]L}}];

  \draw [output] (A) -- (L);

  \foreach \point in {A,B,C,D,G,L}
    \fill [black,opacity=.5] (\point) circle (2pt);

  % \node ...
\end{tikzpicture}
```