The rulercompass TikZ Library

Andrew Stacey

10th December 2013

1 Introduction

The rulercompass TikZ library provides some commands and styles for drawing straight-edge[1] and compass diagrams in TikZ. With this, one can draw diagrams such as that in Example 1.

2 Straight-Edge and Compass Diagrams

To understand the commands of this package, it is necessary to know a bit about the type of diagram involved. A straight-edge and compass diagram consists of a family of points in the plane, a family of straight lines, and a family of circles. The usual rules for constructing these families is to start with a given set of points and apply the following rules:

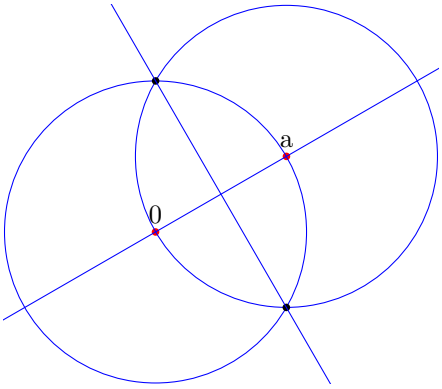
1. Construct a straight line passing through any two points.
2. Construct a circle centred at one point which passes through another point.
3. Place a point at an intersection of two lines, two circles, or a circle and a line.

The usual goal of a straight-edge and compass diagram is to place a point at a particular position given a certain family of initial points. Some examples are:

1. Given three points in the plane, say $0, a, b$, view a and b as vectors from 0 and construct their sum, $a + b$.
2. Given four points in the plane, say $0, 1, a, b$, view a and b as complex numbers with 0 and 1 interpreted in the obvious way, and construct their product, ab.
3. Given two points in the plane, say 0 and 1, construct a regular hexagon with 0 and 1 as adjacent vertices.

[1] The name straightedgecompass was just a bit too long.

```
\begin{tikzpicture}[
  stop jumping,
  constrain
]
\path (0,0) node[name=0,ruler
    compass/point=red,label={0}];
\path (0) ++(30:2) node[ruler
    compass/point=red,label={a}];
\ruler{0}{a}
\compass{0}{a}
\compass{a}{0}
\point{c0a}{ca0}{1}
\point{c0a}{ca0}{2}
\ruler{b}{c}
\end{tikzpicture}
```



Example 1: Constructing the perpendicular bisector between two points

4. Given two points in the plane, say 0 and 1, construct a regular 17–sided polygon with 0 and 1 as adjacent vertices.

Often, a diagram is completed by "inking in" some segments of the lines and circles.

When drawing a straight-edge and compass diagram we need to be able to carry out the three constructions. Whilst none correspond to basic TikZ path operations, they are all possible with the use of some additional libraries, namely the `intersections` and `calc` libraries. These are therefore loaded automatically.

3 Usage

This package comes as a TikZ library so can be loaded in the same way as other TikZ libraries:

```
\usetikzlibrary{rulercompass}
```

This loads in a family of commands and TikZ styles for drawing these diagrams.

3.1 Naming Schemes

The various constructions refer to points or paths already constructed (or given at the start). Therefore, it is necessary to have a naming scheme.

Points are labelled automatically using an internal LaTeX counter, `pointlabels`. The format name can be changed in the same way as the format of any LaTeX counter, by redefining `\thepointlabels`. However, because it is sometimes needed to be able to go back from the label to the value of the counter, it is better to change it by one of the TikZ styles provided. As a default, `rulercompass` uses lowercase letters. Thus the first point will be `a`, the second `b`, and so on. Points are actually coordinates so these can be used as such within the `tikzpicture`.

Since paths are defined in terms of points, the naming scheme for paths is a triplet of tokens or token lists where the first distinguishes between lines and circles and the other two are the two points used to construct the path.

- `rab` is a line (`r` for "ruler") through the points `a` and `b`.

  Note that although `rab` and `rba` are the same line, the two names cannot be used interchangeably. The order of points must be the same as that used in the construction of the line.

- `c{ab}v` is a circle (`c` for "compass") with centre `ab` passing through `v`.

  Here there is no risk of confusion since `cab` and `cba` are distinct circles.

3.2 Primary Commands

There are three primary commands available:

- `\point[style]{<path>}{<path>}{<intersection>}`

  This marks an intersection with a point, using the next label. Some pairs of paths have two intersection points and the `<intersection>` argument is used to choose which one to mark.

- `\ruler[style]{<point>}{<point>}`

  This constructs the line through the two points.

- `\compass[style]{<point>}{<point>}`

  This constructs the circle centred at the first point which passes through the second point.

If the `rulercompass` library detects that `beamer` has been loaded, then all of the above are made overlay-aware. In addition, the `\point` command does a little optimisation: as intersections can be expensive to compute (if there are a lot of them), then it tries to figure out if this particular intersection has already been computed in a previous version of this diagram.

Example 1 contains an example of each of these commands.

3.3 Styles

The primary commands all take an optional argument to which arbitrary TikZ styles can be passed. Each command internally expands to a `\draw` (for `\ruler` and `\compass`) or a `node` on a `\path` (for `\point`) with the optional argument inserted in a suitable place. If this does not provide enough opportunity for control, there are style equivalents of the commands that can be used directly:

- `ruler compass/point=<colour>`

  This marks the current point. Inside the `\point` command, this is invoked inside a `node` after the appropriate intersection has been computed. The `<colour>` is optional. If the node has not already been named (via `name=<name>`) then this gives it the next name as determined by the `pointlabels` counter.

- `ruler compass/ruler={<point>}{<point>}`

  This inserts the line through the points into the current path.

- `ruler compass/compass={<point>}{<point>}`

  This inserts the compass with centre the first point which passes through the second point into the current path.

The initial two commands in Example 1 use the style format for greater flexibility. In the first, the `name=0` key overrides the automatic naming of the point meaning that this point is known as `0` instead of `a`. In both the first and the second, a colour is specified.

There are a variety of styles that can be used to change how the various constructions are rendered:

- `ruler compass/every point`
- `ruler compass/every construction path`

  Encompasses both `ruler` and `compass`.
- `ruler compass/every ruler`
- `ruler compass/every compass`
- `ruler compass/construction in use` and `ruler compass/construction not in use`

  When `beamer` is detected, every time a path is used to compute a point then the current frame is noted. This means that on the second (and subsequent) runs, it is possible to style paths differently depending on whether or not they are still in use.
- `ruler compass/every aux point`

  When a path is computed, then the two points that are used in its construction are designated as "auxiliary points" and are given an extra style. This is mainly of use when stepping through a construction in `beamer`.
- `ruler compass/draft label`

  If in draft mode (determined by the key `ruler compass/draft mode`), then constructed points are labelled. This can be used to style the label.

Some other miscellaneous keys are the following.

- `ruler compass/draft mode=<true|false>`

  This controls a switch to display the labels above the constructed points.
- `ruler compass/ruler length=<length>`

  The default line extends 20cm beyond each point. This changes that.
- `ruler compass/point labels=<arabic|alph|Alph|alphalph|AlphAlph>`

  This changes the style of the point labels. For the last two, the `alphalph` package must be loaded.

  If this library detects that the `alphalph` package has already been loaded then the default is `alphalph`, otherwise the default is `alph`.

  It is better to use this than to redefine `\thepointlabels`.

4 Relative Labels

The system for labelling points has rudimentary support for arithmetic. For this, it is important that TEX know how to reverse the labelling system to get a number back from a label and this is why the key `ruler compass/point labels` should be used to change the labelling format. The operations allowed are addition and

subtraction of a number from a label, together with the use of a period to designate the last marked point. This makes it easier to work on a drawing in parts without having to know exactly how many points are constructed in each part, or to define auxiliary macros to construct some number of extra points.

In Example 2, in the line `\ruler{0}{.}` then the second point is `b`. In the line `\compass{0}{.-1}` then the second point is `a`. In the last line, the second path is the circle with centre `0` which goes through `b`.

5 Bounding Boxes

There are two considerations here relating to bounding boxes. The first is the "infinite" lines. Even though these are not actually infinite, they are constructed to be very long (so that any intersection is almost certain to actually exist). This usually means that they dwarf the actual region of interest. To ensure that the diagram does not get too big, the lines are actually drawn without affecting the bounding box (via the `overlay` key), but that simply means that they can bleed into the surrounding text. Clearly a `\clip` would be useful here, but it would also be nice not to have to compute it by hand.

The second consideration comes when using `beamer` and doing a step-by-step construction. Each piece of the construction adds a new element and thus, potentially, changes the bounding box. This could lead to the picture jumping around on the page. Inserting a `\path` of a suitable size would solve this, but again it would be nice not to have to compute this by hand.

Both of these have the same solution: compute the effective bounding box at the end of the construction and use that information at the start. This involves saving information to the `aux` file and so requires at least two runs.

In the first situation, we clip the picture to its eventual bounding box (possibly slightly enlarged). In the second, we add a rectangle of the eventual bounding box at the end (adding it at the end means that we can first compute the bounding box without it to ensure that our saved bounding box is always up to date).

Once we give ourselves the technology to save the bounding box, we can also think of other ways to use it. One more is given here: the ability to automatically resize a `tikzpicture` to fit inside a given rectangle (preserving the aspect ratio).

In addition, a pseudo-node `enclosing box` is defined which can be used like the `current bounding box` pseudo-node.

The original code for saving and reusing the bounding box comes from the TeX-SX site question How can I fix jumping TikZ pictures in beamer?.

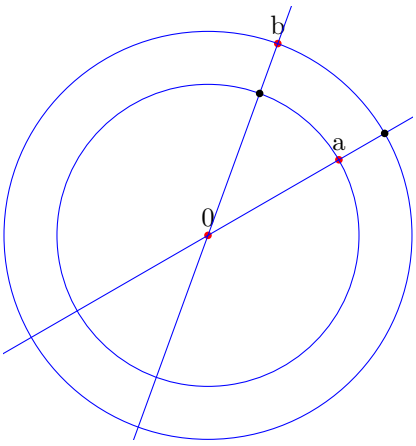
This code provides the following new keys which are designed to be used in the optional argument to the `tikzpicture` environment:

- `stop jumping`

  This installs the code to save the bounding box and also to insert the rectangle at the end of the picture.

- `max size={<width>}{<height>}`

```
\begin{tikzpicture}[
  stop jumping,
  constrain
]
\path (0,0) node[name=0,ruler
    compass/point=red,label={0}];
\path (0) ++(30:2) node[ruler
    compass/point=red,label={a}];
\path (0) ++(70:2.7) node[ruler
    compass/point=red,label={b}];
\ruler{0}{.}
\ruler{0}{a}
\compass{0}{.-1}
\point{r0b}{c0a}{1}
\compass{0}{a+1}
\point{r0a}{c0{a+1}}{1}
\end{tikzpicture}
```



Example 2: Demonstrating relative point labels.

This compares the (saved) bounding box to the given rectangle and then scales down (only down) to ensure that it fits inside the given rectangle.

- `enclosing box/offset=<length>`

  This adds a little extra room around the bounding box. This has an effect in two places: on the computation in the `max size` it ensures that the extra room fits inside the given rectangle, and on the size of the `enclosing box` pseudo-node.

- `constrain`

  If the `enclosing box` pseudo-node is used directly, it should almost always be used with the `overlay` key because otherwise there is potential for the bounding box size to explode.

  This makes it somewhat tricky to use with the `\clip` path constructor as that does its own stuff with the computation of the bounding box. For this reason, the key `constrain` is provided which does a "safe" clip to the enclosing box.

  If this is to be used somewhere other than at the start of the picture, there is the command `\constrain`.

6 Drawing Segments

Drawing a line segment between two points is already adequately provided for by TikZ. Drawing an arc is not, so this library provides a key `centred arc to={<point>}{<point>}` which draws an arc from the current point to the second specified point which is centred at the first specified point. The key `arc flip` can be used to switch which part of the circle is drawn. These keys are for the `arc` path constructor, as in Example 3.

7 Miscellaneous Extras

As a little extra (because I found them useful when using this library), this library also provides the code used in the answers to "Z-level in TikZ" and How can I force TikZ pin angle?. The first provides a per-path and per-node layering system, the second helps with label positioning. The extra keys are:

- `on layer=<layer>`

  This puts the current path on a particular layer.

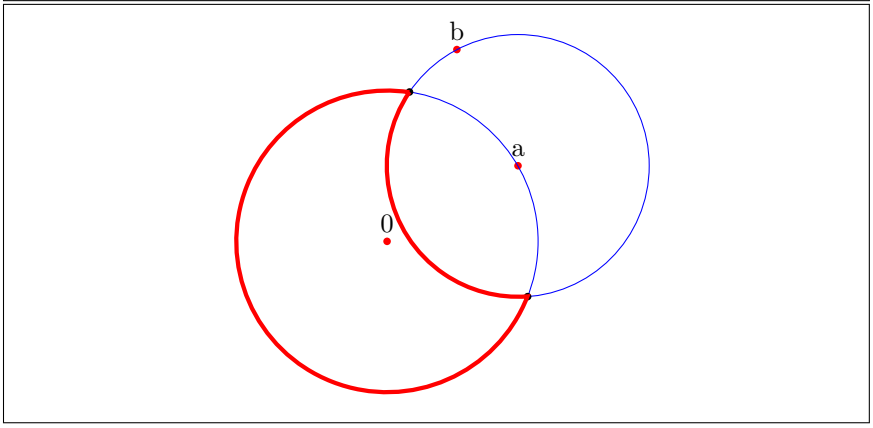
- `node on layer=<layer>`

  This is the same, but for nodes.

- `reset label anchor=<anchor>`

  This turns off TikZ's automatic label anchoring to make it possible to specify a particular anchor to use.

```
\begin{tikzpicture}
\path (0,0) node[name=0,ruler
    compass/point=red,label={0}];
\path (0) ++(30:2) node[ruler
    compass/point=red,label={a}];
\path (0) ++(70:2.7) node[ruler
    compass/point=red,label={b}];
\compass{0}{a}
\compass{a}{b}
\point{c0a}{cab}{1}
\point{c0a}{cab}{2}
\draw[ultra thick,red] (c) arc[centred arc to={a}{d}]
    arc[arc flip,centred arc to={0}{c}];
\end{tikzpicture}
```



Example 3: Drawing an arc.

8 Beamer Examples

The following show some examples using `beamer` to step through. As these are "full screen", the page is used as the bounding box and so the `constrain` key is not needed.

8.1 Addition of Points

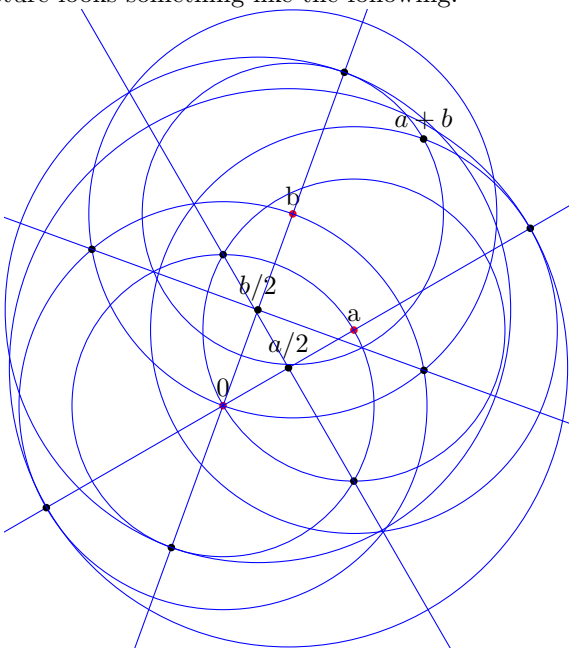
```
\documentclass[
%  handout % use this to see the final construction
]{beamer}
\usepackage{alphalph}
\usepackage{tikz}
\usetikzlibrary{rulercompass}

\pgfmathsetmacro\ptra{2}%{.5 + .25*rand}
\pgfmathsetmacro\ptrb{2.7}%{.5 + .25*rand}
\pgfmathsetmacro\ptaa{30}%180*rand}
\pgfmathsetmacro\ptab{70}%180*rand}

\begin{document}
\begin{frame}[plain]
\hfill%
\begin{tikzpicture}[
    stop jumping,
    max size={\textwidth}{\textheight},
  ]
% Mark two points: one at the centre of the page and one a
    random offset.  These define our unit of length for
    constructions
\path (0,0) node[name=0,ruler compass/point=red,label={0}];
\path (0) ++(\ptaa:\ptra) node[ruler
    compass/point=red,label={a}];
\path (0) ++(\ptab:\ptrb) node[ruler
    compass/point=red,label={b}];
\pause
\ruler<+->{0}{b}
\ruler<+->{0}{a}
\compass<+->{0}{a}
\point<.->{r0b}{c0a}{2}
\compass<+->{0}{b}
\point<.->{r0a}{c0b}{2}
\compass<+->{a}{0}
\point<.->{ca0}{c0a}{1}
\point<.->{ca0}{c0a}{2}
\ruler<+->{e}{f}
```

```
\point<.->[label={[above]\(a/2\)}]{r0a}{ref}{1}
\compass<+->{g}{d}
\point<.->{r0a}{cgd}{1}
\compass<+->{a}{h}
\compass<+->{b}{0}
\point<.->{cb0}{c0b}{1}
\point<.->{cb0}{c0b}{2}
\ruler<+->{i}{j}
\point<.->[label={[above]\(b/2\)}]{r0b}{rij}{1}
\compass<+->{k}{c}
\point<.->{r0b}{ckc}{1}
\compass<+->{b}{l}
\point<.->[fill=red,label={[above]\(a+b\)}]{cbl}{cah}{1}
\end{tikzpicture}%
\hspace*{\fill}%
\end{frame}
\end{document}
```

The final picture looks something like the following.



8.2 Multiplication of Points

```
\documentclass[
%  handout % use this to see the final construction
]{beamer}
%\url{http://tex.stackexchange.com/q/142210/86}
```

```
\usepackage{tikz}
\usetikzlibrary{rulercompass}

\setbeamertemplate{navigation symbols}{}

\pgfmathsetmacro\ptra{2}%{.5 + .25*rand}
\pgfmathsetmacro\ptrb{2.7}%{.5 + .25*rand}
\pgfmathsetmacro\ptaa{30}%180*rand}
\pgfmathsetmacro\ptab{70}%180*rand}

\begin{document}
\begin{frame}[plain]
\hfill
\begin{tikzpicture}[
    stop jumping,
    max size={\textwidth}{\textheight}
  ]
% Mark two points: one at the centre of the page and one a
    random offset.  These define our unit of length for
    constructions
\path (0,0) node[name=0,ruler compass/point=red,label={0}];
\path (1,0) node[name=1,ruler compass/point=red,label={1}];
\path (0) ++(\ptaa:\ptra) node[ruler
   compass/point=red,label={a}];
\path (0) ++(\ptab:\ptrb) node[ruler
   compass/point=red,label={b}];
\pause
\ruler<+->{0}{1}
\ruler<.->{0}{b}
\compass<+->{0}{1}
\point<.->{c01}{r0b}{1}
\compass<+->{1}{a}
\point<.->{c1a}{r01}{1}
\compass<+->{0}{d}
\point<.->{c0d}{r0b}{1}
\point<.->{c0d}{r0b}{2}
\compass<+->{0}{a}
\compass<+->{c}{e}
\point<.->{c0a}{cce}{2}
\point<.->{c0a}{r0b}{2}
\ruler<+->{0}{g}
\compass<+->{0}{b}
\compass<+->{b}{0}
\point<.->{c0b}{cb0}{1}
\point<.->{c0b}{cb0}{2}
\ruler<+->{i}{j}
```

```
\point<.->{r0b}{rij}{1}
\compass<+->{k}{c}
\point<.->{r0b}{ckc}{1}
\compass<+->{0}{l}
\compass<+->{l}{0}
\point<.->{c0l}{cl0}{1}
\point<.->{c0l}{cl0}{2}
\ruler<+->{m}{n}
\point<.->{r0b}{rmn}{1}
\compass<+->{o}{f}
\point<.->{cof}{r0b}{1}
\compass<+->{o}{h}
\point<.->{coh}{r0b}{1}
\compass<+->{l}{q}
\compass<+->{b}{p}
\point<.->{clq}{cbp}{2}
\ruler<+->{b}{r}
\point<+->[fill=red,label={[above]\(ab\)}]{rbr}{r0g}{1}
\end{tikzpicture}%
\hspace*{\fill}%
\end{frame}
\end{document}
```

A comparison of the `beamer` version with the following picture will show the wisdom of a step-by-step approach to these drawings.