The rulercompass package: code

Andrew Stacey
stacey@math.ntnu.no

December 11, 2013

1 Implementation

Load in useful tikzlibraries.

```
\usetikzlibrary{intersections,calc}
```

Are we in draft mode (so display point labels)?

```
\newif\ifrc@draft
```

Is beamer loaded?

```
\newif\ifrc@beamer
```

When drawing arc segments, do we flip the segment?

```
\newif\ifrc@fliparc
```

A picture id that stays the same on beamer frames to make it easier to compare bounding boxes and other information of the "same" tikzpicture.

```
\newcount\rc@picture@id
```

Counter for our point labels.

```
\newcounter{pointlabels}
```

Set the beamer boolean.

```
\@ifclassloaded{beamer}{%
  \rc@beamertrue
}{}
```

Internal separator for the path-naming scheme

```
\def\rc@pathsep{@}
```

Are we running under beamer?

```
\ifrc@beamer
```

Make our counters reset on frames

```
\resetcounteronoverlays{pointlabels}
\resetcountonoverlays{rc@picture@id}
```

Define an overlay-aware style

```
\tikzset{
  alt if exist/.code args={#1#2#3}{%
    \@ifundefined{path@\the\rc@picture@id @#1}{%
      \pgfkeysalso{#2}%
    }{
      \alt<.-\csname path@\the\rc@picture@id @#1\endcsname>{%
        \pgfkeysalso{#2}%
      }{
        \pgfkeysalso{#3}%
      }%
    }%
  },
```

Save a path when it is used to compute a point

```
  intersection/save/.code={%
    \only<.>{%
      \begingroup
      \tikz@intersect@path@names@parse#1\tikz@stop
      \protected@write\pgfutil@auxout{}{%
        \string\global\string\@namedef{path@\the\rc@picture@id @\tikz@intersect@path@a}{\the\b
        \string\global\string\@namedef{path@\the\rc@picture@id @\tikz@intersect@path@b}{\the\b
      }%
    \endgroup
    }%
  }
}
```

Define overlay-aware versions of the main macros.

```
\newcommand<>\compass[3][]{%
  \draw#4[#1,ruler compass/compass={#2}{#3}];
}

\newcommand<>\ruler[3][]{%
  \draw#4[#1,ruler compass/ruler={#2}{#3}];
}
```

The `\point` macro is doubly overlay aware. If the intersection has already been computed (say, on another slide of the same picture), reuse it.

```
\newcommand<>\point[4][]{%
  {%
    \advance\c@pointlabels by 1\relax
    \xdef\rc@temp{\thepointlabels}%
  }%
  \edef\rc@tempa{\the\rc@picture@id}%
  \expandafter\ifx\csname rc@id@\rc@temp\endcsname\rc@tempa
    \path#5 (\rc@temp) node[ruler compass/point,#1] {};
  \else
    \path#5 [name intersections={use=#2 and #3}] (intersection-#4) node[ruler compass/point,#1
  \fi
}
```

Now for the non-beamer versions

```
\else
```

The overlay-aware style defaults to the first option.

```
\tikzset{
  alt if exist/.code args={#1#2#3}{%
    \pgfkeysalso{#2}%
  },
```

Saving this means we don't have to have two versions of a more complicated bit of code.

```
  intersection/save/.code={%
    \begingroup
    \tikz@intersect@path@names@parse#1\tikz@stop
    \protected@write\pgfutil@auxout{}{%
        \string\global\string\@namedef{path@\the\rc@picture@id @\tikz@intersect@path@a}{\thepag
        \string\global\string\@namedef{path@\the\rc@picture@id @\tikz@intersect@path@b}{\thepag
    }%
    \endgroup
  }
}
```

Non-overlay aware versions of the primary commands.

```
\newcommand\compass[3][]{%
  \draw [#1,ruler compass/compass={#2}{#3}];
}

\newcommand\ruler[3][]{%
  \draw [#1,ruler compass/ruler={#2}{#3}];
}

\newcommand\point[4][]{%
  \path [name intersections={use=#2 and #3}] (intersection-#4) node[ruler compass/point,#1] {}
}
```

End of non-beamer specific section

```
\fi
```

The following macros process the path/point specifications. The first looks to see if the first character is a period.

```
\def\rc@processpt#1{%
  \pgfutil@ifnextchar.{\rc@processpt@@#1}{\rc@processpt@#1}}
```

Nope, so now look for a plus or a minus.

```
\def\rc@processpt@#1#2\pgf@stop{%
  \pgfutil@in@+{#2}%
  \ifpgfutil@in@
   \let\@next=\rc@processpt@plus
  \else
   \pgfutil@in@-{#2}%
   \ifpgfutil@in@
    \let\@next=\rc@processpt@minus
   \else
```

```
    \let\@next=\rc@processpt@bare
   \fi
  \fi
  \@next#1#2\pgf@stop
}
```

Okay, we got a period. That means the current point, possibly with an offset.

```
\def\rc@processpt@@#1.#2\pgf@stop{%
  \def\rc@temp{#2}%
  \ifx\rc@temp\pgfutil@empty
  \edef\rc@temp{\thepointlabels}%
  \else
  \pgfmathparse{int(\the\c@pointlabels + #2)}%
  \begingroup
  \c@pointlabels=\pgfmathresult\relax
  \xdef\rc@temp{\thepointlabels}%
  \endgroup
  \fi
  \let#1\rc@temp
}
```

We need to add an offset to the given label, so need to compute the index of the label.

```
\def\rc@processpt@plus#1#2+#3\pgf@stop{%
  \pgfmathsetmacro\rc@temp{0}%
  \rc@reverse#2\pgf@stop%
  \pgfmathparse{int(\rc@temp + #3)}%
  \begingroup
  \c@pointlabels=\pgfmathresult\relax
  \xdef\rc@temp{\thepointlabels}%
  \endgroup
  \let#1\rc@temp
}
```

Same, but with a minus.

```
\def\rc@processpt@minus#1#2-#3\pgf@stop{%
  \pgfmathsetmacro\rc@temp{0}%
  \rc@reverse#2\pgf@stop%
  \pgfmathparse{int(\rc@temp - #3)}%
  \begingroup
  \c@pointlabels=\pgfmathresult\relax
  \xdef\rc@temp{\thepointlabels}%
  \endgroup
  \let#1\rc@temp
}
```

Simplest case.

```
\def\rc@processpt@bare#1#2\pgf@stop{%
  \def#1{#2}}
```

Reverse lookup the counter value from its displayed format.

```
\def\rc@reverse#1{%
  \ifx#1\pgf@stop
```

```
    \let\@next=\pgfutil@empty
  \else
    \let\@next=\rc@reverse
    \pgfutil@tempcnta=`#1\relax
    \pgfmathsetmacro\rc@temp{\rc@temp * \rc@factor + \the\pgfutil@tempcnta - \rc@initial}%
  \fi
  \@next}
```

Compute the actual path name from the three token (lists).

```
\def\rc@parsepath#1#2#3#4\pgf@stop{%
  \rc@processpt\rc@tempa#3\pgf@stop
  \rc@processpt\rc@tempb#4\pgf@stop
  \xdef#1{#2\rc@pathsep\rc@tempa\rc@pathsep\rc@tempb}%
}
```

Now we install all our styles

```
\tikzset{
```

Code which initialises our counters at the start of a picture.

```
  every picture/.append style={
    ruler compass/at begin picture
  },
```

Passes the paths to the intersection but also takes note of their use.

```
  intersection/use/.code args={#1 and #2}{%
    \rc@parsepath\rc@pta#1\pgf@stop
    \rc@parsepath\rc@ptb#2\pgf@stop
    \pgfkeysalso{
      of={\rc@pta} and \rc@ptb,
      save={\rc@pta} and \rc@ptb,
    }
  },
```

`to path` for a circle through a given point.

```
  circle through/.style={
    to path={
      \pgfextra{
        \tikz@scan@one@point\pgfutil@firstofone(\tikztostart)\relax
        \pgf@xa=\pgf@x
        \pgf@ya=\pgf@y
        \tikz@scan@one@point\pgfutil@firstofone(\tikztotarget)\relax
        \pgfmathsetmacro\rc@radius{veclen(\pgf@x - \pgf@xa,\pgf@y - \pgf@ya)}%
      }
      circle[radius=\rc@radius pt] (\tikztotarget)
    }
  },
```

Flip an arc when drawing a segment.

```
  arc flip/.is if=rc@fliparc,
```

Arc from one point to another centred on specified point.

```
  centred arc to/.code 2 args={%
    \tikz@scan@one@point\pgfutil@firstofone(#1)\relax
    \pgfmathsetmacro\rc@radius{veclen(\tikz@lastx-\pgf@x,\tikz@lasty-\pgf@y)}%
    \pgfmathsetmacro\rc@sangle{atan2(\tikz@lasty - \pgf@y,\tikz@lastx - \pgf@x)}%
    \pgf@xa=\pgf@x
    \pgf@ya=\pgf@y
    \tikz@scan@one@point\pgfutil@firstofone(#2)\relax
    \pgfmathsetmacro\rc@eangle{atan2(\pgf@y - \pgf@ya,\pgf@x - \pgf@xa)}%
    \pgfmathsetmacro\rc@eangle{\rc@eangle < \rc@sangle ? \rc@eangle +       360 : \rc@eangle}%
    \ifrc@fliparc
    \pgfmathsetmacro\rc@eangle{\rc@eangle - 360}%
    \fi
    \pgfkeysalso{start angle=\rc@sangle, end angle=\rc@eangle, radius=\rc@radius pt}
  },
```

Most of our keys are in this family.

```
  ruler compass/.is family,
  ruler compass/.cd,
```

Draft mode displays the labels.

```
  draft mode/.is if=rc@draft,
%  beamer mode/.is if=rc@beamer,
```

This is the actual code for resetting the counters.

```
  at begin picture/.style={
    execute at begin picture={
      \global\advance\rc@picture@id by 1\relax
      \setcounter{pointlabels}{0}%
    },
  },
```

Styling the point labels.

```
  point labels/.is choice,
  point labels/arabic/.code={%
    \renewcommand\thepointlabels{\@arabic \c@pointlabels}%
    \def\rc@initial{48}%
    \def\rc@factor{10}%
  },
  point labels/alph/.code={%
    \renewcommand\thepointlabels{\@alph \c@pointlabels}%
    \def\rc@initial{96}%
    \def\rc@factor{26}%
  },
  point labels/Alph/.code={%
    \renewcommand\thepointlabels{\@Alph \c@pointlabels}%
    \def\rc@initial{64}%
    \def\rc@factor{26}%
  },
```

Styling the point labels with the `alphalph` package.

```
  point labels/alphalph/.code={%
    \@ifundefined{alphalph}{%
      \message{The "alphalph" option only works if the 'alphalph' package has been loaded (usi
      \renewcommand\thepointlabels{\@alph \c@pointlabels}%
      \def\rc@initial{96}%
      \def\rc@factor{26}%
    }{%
      \renewcommand\thepointlabels{\alphalph \c@pointlabels}%
      \def\rc@initial{96}%
      \def\rc@factor{26}%
    }%
  },
  point labels/AlphAlph/.code={%
    \@ifundefined{alphalph}{%
      \message{The "AlphAlph" option only works if the 'alphalph' package has been loaded (usi
      \renewcommand\thepointlabels{\@Alph \c@pointlabels}%
      \def\rc@initial{64}%
      \def\rc@factor{26}%
    }{%
      \renewcommand\thepointlabels{\AlphAlph \c@pointlabels}%
      \def\rc@initial{64}%
      \def\rc@factor{26}%
    }%
  },
```

Styles for if the path is still in use or not.

```
  construction in use/.style={
    draw=blue
  },
  construction not in use/.style={
    draw=gray!75,
  },
```

Code for drawing a circle.

```
  compass/.code 2 args={%
    \rc@processpt\rc@pta#1\pgf@stop
    \rc@processpt\rc@ptb#2\pgf@stop
    \pgfkeysalso{ruler compass/compass aux={\rc@pta}{\rc@ptb}}%
  },
  compass aux/.style 2 args={
    alt if exist={c\rc@pathsep#1\rc@pathsep#2}{
      ruler compass/construction in use/.try,
    }{
      ruler compass/construction not in use/.try
    },
    name path=c\rc@pathsep#1\rc@pathsep#2,
    ruler compass/every construction path/.try,
    ruler compass/every compass/.try,
    insert path={
```

```
      (#1) to[circle through] (#2)
      node[name=c\rc@pathsep#1\rc@pathsep#2\rc@pathsep centre,ruler compass/aux point={#1}] {}
      node[name=c\rc@pathsep#1\rc@pathsep#2\rc@pathsep rim,ruler compass/aux point={#2}] {}
    },
  },
```

Code for drawing a straight line.

```
  ruler/.code 2 args={%
    \rc@processpt\rc@pta#1\pgf@stop
    \rc@processpt\rc@ptb#2\pgf@stop
    \pgfkeysalso{ruler compass/ruler aux={\rc@pta}{\rc@ptb}}%
  },
  ruler aux/.style 2 args={
    overlay,
    alt if exist={r\rc@pathsep#1\rc@pathsep#2}{
      ruler compass/construction in use/.try
    }{
      ruler compass/construction not in use/.try
    },
    name path=r\rc@pathsep#1\rc@pathsep#2,
    ruler compass/every construction path/.try,
    ruler compass/every ruler/.try,
    insert path={
      ($(#1)!\pgfkeysvalueof{/tikz/ruler compass/ruler length}!(#2)$) -- ($(#2)!\pgfkeysvalueo
      node[name=r\rc@pathsep#1\rc@pathsep#2\rc@pathsep start,ruler compass/aux point={#1}] {}
      node[name=r\rc@pathsep#1\rc@pathsep#2\rc@pathsep end,ruler compass/aux point={#2}] {}
    },
  },
```

Sets the ruler length.

```
  ruler length/.initial=20,
```

Auxiliary point style.

```
  aux point/.style={
    node contents/.try={},
    at={(#1)},
    ruler compass/every aux point/.try
  },
```

Draws a point as a coordinate with another node for styling.

```
  point/.style={
    ruler compass/name it,
    coordinate,
    node contents/.try={},
    insert path={
      node[
        fill=#1,
        circle,
        minimum width=1mm,
        inner sep=0mm,
        reset label anchor,
```

```
        anchor=center,
        node contents/.try={},
        ruler compass/every point/.try,
      ] {}
    }
  },
```

Forces a point to be named.

```
  name it/.code={%
    \ifx\tikz@fig@name\pgfutil@empty
    \stepcounter{pointlabels}%
    \pgfkeysalso{name=\thepointlabels}%
    \expandafter\xdef\csname rc@id@\thepointlabels\endcsname{\the\rc@picture@id}%
    \ifrc@draft
    \pgfkeysalso{label={[ruler compass/draft label/.try]\thepointlabels}}%
    \fi
    \fi
  },
  point/.default=black,
}
```

Sets the initial scale and offset of the enclosing box.

```
\def\rc@scale{1}
\def\rc@offset{0pt}
```

Sets the bounding box from the auxiliary file.

```
\def\jump@setbb#1#2#3{%
  \@ifundefined{jump@#1@maxbb}{%
    \expandafter\gdef\csname jump@#1@maxbb\endcsname{#3}%
  }{%
    \csname jump@#1@maxbb\endcsname
    \pgf@xa=\pgf@x
    \pgf@ya=\pgf@y
    #3
    \pgfmathsetlength\pgf@x{max(\pgf@x,\pgf@xa)}%
    \pgfmathsetlength\pgf@y{max(\pgf@y,\pgf@ya)}%
    \expandafter\xdef\csname jump@#1@maxbb\endcsname{\noexpand\pgfpoint{\the\pgf@x}{\the\pgf@y
  }
  \@ifundefined{jump@#1@minbb}{%
    \expandafter\gdef\csname jump@#1@minbb\endcsname{#2}%
  }{%
    \csname jump@#1@minbb\endcsname
    \pgf@xa=\pgf@x
    \pgf@ya=\pgf@y
    #2
    \pgfmathsetlength\pgf@x{min(\pgf@x,\pgf@xa)}%
    \pgfmathsetlength\pgf@y{min(\pgf@y,\pgf@ya)}%
    \expandafter\xdef\csname jump@#1@minbb\endcsname{\noexpand\pgfpoint{\the\pgf@x}{\the\pgf@y
  }
}
```

Installs the code to save the bounding box.

```
\tikzset{
  stop jumping/.style={
    execute at end picture={%
      \pgfmathsetlength\pgf@xa{\pgf@picminx/\rc@scale}%
      \pgfmathsetlength\pgf@ya{\pgf@picminy/\rc@scale}%
      \pgfmathsetlength\pgf@xb{\pgf@picmaxx/\rc@scale}%
      \pgfmathsetlength\pgf@yb{\pgf@picmaxy/\rc@scale}%
      \immediate\write\pgfutil@auxout{%
        \noexpand\jump@setbb{\the\rc@picture@id}{\noexpand\pgfpoint{\the\pgf@xa}{\the\pgf@ya}}
      },
      \pgf@x=\pgf@picminx
      \pgf@y=\pgf@picminy
      \csname jump@\the\rc@picture@id @minbb\endcsname
      \pgf@xa=\pgf@x
      \pgf@ya=\pgf@y
      \pgf@x=\pgf@picmaxx
      \pgf@y=\pgf@picmaxy
      \csname jump@\the\rc@picture@id @maxbb\endcsname
      \edef\rc@temp{\noexpand\path (\the\pgf@xa - \rc@offset,\the\pgf@ya - \rc@offset) -- (\the
      \rc@temp
    },
  },
```

Scales the picture to fit inside a given rectangle.

```
  max size/.code 2 args={%
    \pgfutil@ifundefined{jump@\the\rc@picture@id @maxbb}{}{%
      \csname jump@\the\rc@picture@id @maxbb\endcsname
      \pgf@xa=\pgf@x
      \pgf@ya=\pgf@y
      \csname jump@\the\rc@picture@id @minbb\endcsname
      \advance\pgf@xa by -\pgf@x
      \advance\pgf@ya by -\pgf@y
      \advance\pgf@xa by \rc@offset
      \advance\pgf@xa by \rc@offset
      \advance\pgf@ya by \rc@offset
      \advance\pgf@ya by \rc@offset
      \pgfmathsetmacro\rc@xratio{\pgf@xa > #1 ? \pgf@xa/#1 : 1}%
      \pgfmathsetmacro\rc@yratio{\pgf@ya > #2 ? \pgf@ya/#2 : 1}%
      \pgfmathsetmacro\rc@scale{1/max(\rc@xratio,\rc@yratio)}%
      \pgfkeysalso{scale=\rc@scale}%
    }
  },
  enclosing box/offset/.store in=\rc@offset,
  constrain/.style={
    execute at begin picture=\constrain
  }
}
```

Defines a clip to the enclosing box.

```
\newcommand\constrain{%
  \pgfutil@ifundefined{jump@\the\rc@picture@id @minbb}{}{%
    \csname jump@\the\rc@picture@id @minbb\endcsname
    \pgfmathsetlength\pgf@xa{\pgf@x-\rc@offset}%
    \pgfmathsetlength\pgf@ya{\pgf@y-\rc@offset}%
    \csname jump@\the\rc@picture@id @maxbb\endcsname
    \pgfmathsetlength\pgf@xb{\pgf@x+\rc@offset}%
    \pgfmathsetlength\pgf@yb{\pgf@y+\rc@offset}%
    \edef\rc@temp{\noexpand\clip (\the\pgf@xa,\the\pgf@ya) rectangle (\the\pgf@xb,\the\pgf@yb)
    \ifpgf@relevantforpicturesize
      \pgf@relevantforpicturesizefalse
      \rc@temp
      \pgf@relevantforpicturesizetrue
    \else
      \rc@temp
    \fi
  }%
}
```

Defines the enclosing box node.

```
\expandafter\def\csname pgf@sh@ns@enclosing box\endcsname{rectangle}
\expandafter\def\csname pgf@sh@np@enclosing box\endcsname{%
  \pgfutil@ifundefined{jump@\the\rc@picture@id @minbb}{%
    \def\southwest{\pgfqpoint{\pgf@picminx}{\pgf@picminy}}%
    \def\northeast{\pgfqpoint{\pgf@picmaxx}{\pgf@picmaxy}}%
}{%
    \csname jump@\the\rc@picture@id @minbb\endcsname
    \pgfmathsetlength\pgf@xa{\pgf@x-\rc@offset}%
    \pgfmathsetlength\pgf@ya{\pgf@y-\rc@offset}%
    \edef\southwest{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}%
    \csname jump@\the\rc@picture@id @maxbb\endcsname
    \pgfmathsetlength\pgf@xb{\pgf@x+\rc@offset}%
    \pgfmathsetlength\pgf@yb{\pgf@y+\rc@offset}%
    \edef\northeast{\noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yb}}%
  }%
}
\expandafter\def\csname pgf@sh@nt@enclosing box\endcsname{{1}{0}{0}{1}{0pt}{0pt}}
\expandafter\def\csname pgf@sh@pi@enclosing box\endcsname{\pgfpictureid}


```

Defines the layer code for individual paths and nodes.

```
\tikzset{
  on layer/.code={
    \pgfonlayer{#1}\begingroup
    \aftergroup\endpgfonlayer
    \aftergroup\endgroup
  },
  node on layer/.code={
    \gdef\node@@on@layer{%
```

```
      \setbox\tikz@tempbox=\hbox\bgroup\pgfonlayer{#1}\unhbox\tikz@tempbox\endpgfonlayer\egrou
    \aftergroup\node@on@layer
  },
  reset label anchor/.code={%
    \let\tikz@auto@anchor=\pgfutil@empty
    \def\tikz@anchor{#1}
  },
  reset label anchor/.default=center
}
\def\node@on@layer{\aftergroup\node@@on@layer}
```

Sets the point label style.

```
\@ifpackageloaded{alphalph}%
{
  \tikzset{ruler compass/point labels=alphalph}
}
{
  \tikzset{ruler compass/point labels=alph}
}

\endinput
```