

The triangletools package

Vu Van Dung

May 9, 2020

Contents

1	Introduction	1
2	Loading the package	2
3	User interface	2
3.1	Accessing the keys	2
3.2	Circles associated with triangle centers	2
3.3	Triangle centers	3
3.4	Other utilities	5
3.4.1	Line tools	5
3.4.2	The barycentric coordinate system	8
3.4.3	Distance-finding utility	9
3.5	Customization	9
4	Implementation	9
4.1	The main package file	10
4.2	Errors and warnings	10
4.3	The backend layer	11
4.3.1	The line tools utility	11
4.3.2	The barycentric coordinate system utility	14
4.3.3	Distance-finding utility	16
4.4	Construction of triangle centers	17
4.4.1	X_1 – The incenter	17
4.4.2	X_2 – The centroid	18
4.4.3	X_3 – The circumcenter	18
4.4.4	X_4 – The orthocenter	20
4.4.5	X_5 – The nine-point center	20
4.4.6	X_6 – The symmedian point	20
4.4.7	X_7 – The Gergonne point	21
4.4.8	X_8 – The Nagel point	21
4.4.9	X_9 – The <i>mittenpunkt</i>	21
4.4.10	X_{10} – The Spieker point	22
4.5	The frontend layer	22
	Index	25

1 Introduction

This package aims to help you construct special points in a triangle directly in a short and easy way. Using this package, you can construct most important points listed in Clark Kimberling's Encyclopedia of Triangle Centers (ETC). Currently, all points numbered from X_1 and X_{10} , as well as the excenter, are supported; however with other utilities in this package (see Section 3.4) and a bit of knowledge in geometry and expl3 programming, you can construct even more.

2 Loading the package

This package can be loaded as usual.

```
1 \usepackage{triangletools}
```

It will load TikZ and expl3 automatically.

3 User interface

The user interface of this package, including that of the utilities, is provided as pgf keys under the tree `/tikz/triangletools`.

Note that, in the following sections, a *coordinate* means a *named* TikZ coordinate. That is, in the following example,

```
1 \begin{tikzpicture}
2   \draw (0,0) -- (3,0) coordinate (a);
3 \end{tikzpicture}
```

`a` is a named coordinate, while `0,0` or `3,0` are *not* named coordinates. The current implementation of this package only allows named coordinates in the user interface. It is like the `angles` TikZ library.

3.1 Accessing the keys

```
trt /tikz/trt={\keys}
```

It executes *keys* with the key path set to `/tikz/triangletools`, which is the main key tree of this package.

This key is used to access all other keys in the user interface.

3.2 Circles associated with triangle centers

`\trtradius`

Some points, for example the incenter and the circumcenter, are associated with some special circles. If the requested point is associated with a circle, this macro stores the radius of that circle, in points (pt).

This macro is assigned *globally* every time a point is requested. Therefore, it stores the radius related to the last point that has a circle. So beware that while it always gives you some values once you have drawn such points, that value might not be what you want. It is recommended to use this macro *immediately after* the execution of triangle center keys.

In Section 3.3, if a point has a `\trtradius` associated to it, the circle will be drawn in the code example. Currently X_1 , the excenter, X_3 , X_5 and X_{10} can change the value of `\trtradius`.

If the macro is used before any center with a circle is constructed, an error message will be issued.

3.3 Triangle centers

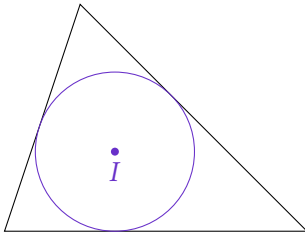
`incenter`

`\trt_sp_incenter:nnnn`

```
/tikz/triangletools/incenter=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)
\trt_sp_incenter:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}
```

Find the incenter X_1 of the triangle joining TikZ coordinates $\langle coor 1 \rangle$, $\langle coor 2 \rangle$ and $\langle coor 3 \rangle$. The incenter is saved to TikZ coordinate $\langle name \rangle$.

If you use the key (why do you use the function anyway), $\langle name \rangle$ is set to `trt output` by default. You can change that using output name, see Section 3.5.



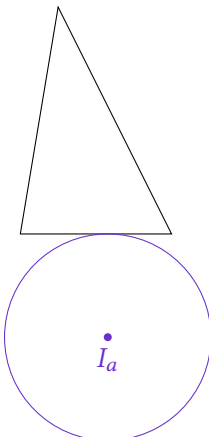
```
1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3     (0,0) coordinate (b) --
4     (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={incenter=(a)(b)(c)}] (trt output)
6     circle[radius=1.5pt] node[below] {$I$};
7   \draw[maincolor] (trt output) circle (\trtradius);
8 \end{tikzpicture}
```

`excenter`

`\trt_sp_excenter:nnnn`

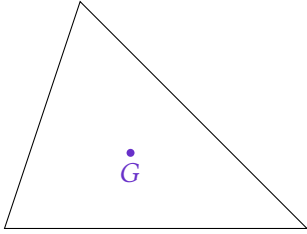
```
/tikz/triangletools/excenter=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)
\trt_sp_excenter:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}
```

Find the excenter of the triangle. The returned point will be on the internal angular bisector at $\langle coor 1 \rangle$. Note that the order matters: `excenter=(a)(b)(c)` is *different* from `excenter=(b)(a)(c)`.



```
1 \begin{tikzpicture}
2   \draw (.5,3) coordinate (a) --
3     (0 ,0) coordinate (b) --
4     (2 ,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={excenter=(a)(b)(c)}] (trt output)
6     circle[radius=1.5pt] node[below] {$I_a$};
7   \draw[maincolor] (trt output) circle (\trtradius);
8 \end{tikzpicture}
```

centroid
 $\backslash\mathrm{trt_sp_centroid:nnnn}$

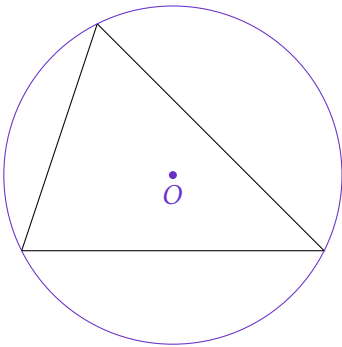


```
/tikz/triangletools/centroid=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)
\trt_sp_centroid:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}
```

Find the centroid X_2 of the triangle.

```
1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={centroid=(a)(b)(c)}] (trt output)
6     circle[radius=1.5pt] node[below] {$G$};
7 \end{tikzpicture}
```

circumcenter
 $\backslash\mathrm{trt_sp_circumcenter:nnnn}$

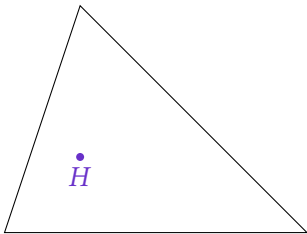


```
/tikz/triangletools/circumcenter=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)
\trt_sp_circumcenter:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}
```

Find the circumcenter X_3 of the triangle.

```
1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={circumcenter=(a)(b)(c)}] (trt output)
6     circle[radius=1.5pt] node[below] {$O$};
7   \draw[maincolor] (trt output) circle (\trtradius);
8 \end{tikzpicture}
```

orthocenter
 $\backslash\mathrm{trt_sp_orthocenter:nnnn}$



```
/tikz/triangletools/orthocenter=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)
\trt_sp_orthocenter:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}
```

Find the orthocenter X_4 of the triangle.

```
1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={orthocenter=(a)(b)(c)}] (trt output)
6     circle[radius=1.5pt] node[below] {$H$};
7 \end{tikzpicture}
```

triangle_center

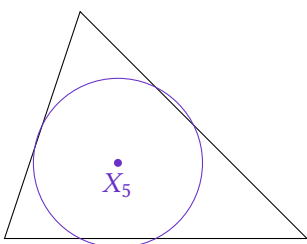
```
/tikz/triangletools/triangle center=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)(\langle index\rangle)
```

Find the point $X_{\langle index \rangle}$ of the triangle. Currently $\langle index \rangle$ can be any integer between and including 1 and 10.

$\backslash\mathrm{trt_sp_ninepointcenter:nnnn}$

```
/tikz/triangletools/triangle center=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)(5)
\trt_sp_ninepointcenter:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}
```

Find the nine-point center X_5 of the triangle.

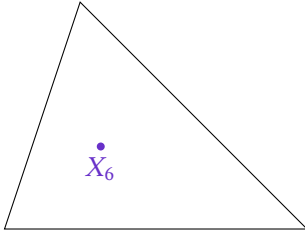


```
1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={triangle center=(a)(b)(c)(5)}] (trt output)
6     circle[radius=1.5pt] node[below] {$X_5$};
7   \draw[maincolor] (trt output) circle (\trtradius);
8 \end{tikzpicture}
```

`\trt_sp_symmedian:nnnn`

`/tikz/triangletools/triangle center=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)(6)`
`\trt_sp_symmedian:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}`

Find the symmedian point X_6 (aka. the Lemoine point or Grebe point) of the triangle.

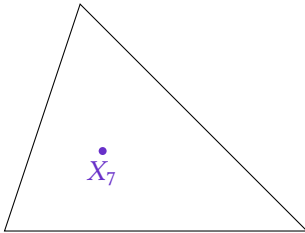


```
1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={triangle center=(a)(b)(c)(6)}] (trt output)
6     circle[radius=1.5pt] node[below] {$X_6$};
7 \end{tikzpicture}
```

`\trt_sp_gergonne:nnnn`

`/tikz/triangletools/triangle center=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)(7)`
`\trt_sp_gergonne:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}`

Find the Gergonne point X_7 of the triangle.

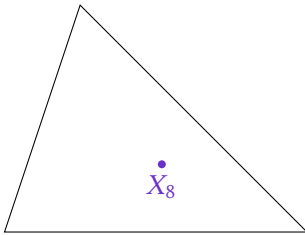


```
1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={triangle center=(a)(b)(c)(7)}] (trt output)
6     circle[radius=1.5pt] node[below] {$X_7$};
7 \end{tikzpicture}
```

`\trt_sp_nagel:nnnn`

`/tikz/triangletools/triangle center=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)(8)`
`\trt_sp_nagel:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}`

Find the Nagel point X_8 of the triangle.

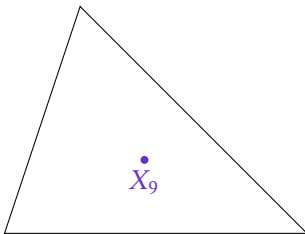


```
1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={triangle center=(a)(b)(c)(8)}] (trt output)
6     circle[radius=1.5pt] node[below] {$X_8$};
7 \end{tikzpicture}
```

`\trt_sp_mittenpunkt:nnnn`

`/tikz/triangletools/triangle center=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)(9)`
`\trt_sp_mittenpunkt:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}`

Find the *mittenpunkt* X_9 of the triangle.

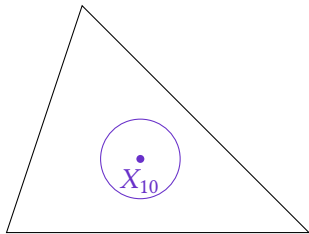


```
1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={triangle center=(a)(b)(c)(9)}] (trt output)
6     circle[radius=1.5pt] node[below] {$X_9$};
7 \end{tikzpicture}
```

`\trt_sp_spieker:nnnn`

`/tikz/triangletools/triangle center=(\langle coor 1\rangle)(\langle coor 2\rangle)(\langle coor 3\rangle)(10)`
`\trt_sp_spieker:nnnn {\langle coor 1\rangle}{\langle coor 2\rangle}{\langle coor 3\rangle}{\langle name\rangle}`

Find the Spieker center X_{10} of the triangle.



```

1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3     (0,0) coordinate (b) --
4     (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={triangle center=(a)(b)(c)(10)}] (trt output)
6     circle[radius=1.5pt] node[below] {$X_{10}$};
7   \draw[maincolor] (trt output) circle (\trradius);
8 \end{tikzpicture}

```

3.4 Other utilities

3.4.1 Line tools

The line tools utility can help you play with some (very basic) operations related to lines.

intersection

/tikz/triangletools/intersection=($\langle coor 1 \rangle$)($\langle coor 2 \rangle$)--($\langle coor 3 \rangle$)($\langle coor 4 \rangle$)

There are two lines, the first joins $\langle coor 1 \rangle$ and $\langle coor 2 \rangle$, and the other joins $\langle coor 3 \rangle$ and $\langle coor 4 \rangle$. This finds the intersection of these lines.

If the two lines are parallel, trt output is set to (0,0), and the package will report a warning.



```

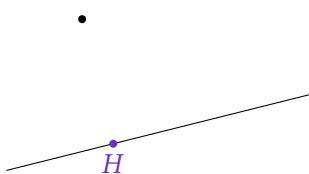
1 \begin{tikzpicture}
2   \draw (0,0) coordinate (a) -- (0,1) coordinate (b);
3   \draw (1,0) coordinate (c) -- (.5,1) coordinate (d);
4   \coordinate (e) at (1,1);
5   \fill[maincolor,trt={intersection=(a)(b)--(c)(d)}] (trt output)
6     circle (1.5pt) node[above] {$I$};
7   \fill[maincolor,trt={intersection=(a)(b)--(c)(e)}] (trt output)
8     circle (1.5pt) node[left] {$J$};
9 \end{tikzpicture}

```

foot_of_perpendicular

/tikz/triangletools/foot of perpendicular=($\langle coor 1 \rangle$)--($\langle coor 2 \rangle$)($\langle coor 3 \rangle$)

Find the foot of perpendicular of point $\langle coor 1 \rangle$ to the line joining $\langle coor 2 \rangle$ and $\langle coor 3 \rangle$.



```

1 \begin{tikzpicture}
2   \draw (0,0) coordinate (b) -- (4,1) coordinate (c);
3   \fill (1,2) circle (1.5pt) coordinate (a);
4   \fill[maincolor,trt={foot of perpendicular=(a)--(b)(c)}] (trt output)
5     circle (1.5pt) node[below] {$H$};
6 \end{tikzpicture}

```

You can do much more using these expl3 functions.

\trt_lt_get_line_equation:nnNNN

\trt_lt_get_line_equation:nnNNN{ $\langle coor 1 \rangle$ }{ $\langle coor 2 \rangle$ }{ $\langle a \rangle$ }{ $\langle b \rangle$ }{ $\langle c \rangle$ }

Find the equation of the line joining $\langle coor 1 \rangle$ and $\langle coor 2 \rangle$, in the form of $ax + by = c$. The l3fp local variables $\langle a \rangle$, $\langle b \rangle$ and $\langle c \rangle$ will be set accordingly.

Note that for any pair of points $\langle coor 1 \rangle$ and $\langle coor 2 \rangle$, there are infinitely many solutions for $\langle a \rangle$, $\langle b \rangle$ and $\langle c \rangle$. This function will produce one of such solution. While the solution is likely to be the simplest of all possible ones, this is not guaranteed.

$$1x + -3y = 1$$

Actual values:

$a = 0.9999995650601154$

$b = -2.999998695180348$

$c = 0.99999913012042$

```

1 \begin{tikzpicture}
2   \coordinate (a) at (1,0);
3   \coordinate (b) at (4,1);
4   \ExplSyntaxOn
5   \fp_new:N \l_foo_tmpa_fp
6   \fp_new:N \l_foo_tmpb_fp
7   \fp_new:N \l_foo_tmpc_fp
8   \trt_lt_get_line_equation:nnNNN {a} {b}
9     \l_foo_tmpa_fp \l_foo_tmpb_fp \l_foo_tmpc_fp
10  \def \resultequation {
11    $\fp_eval:n {round (\l_foo_tmpa_fp / 1cm)}x +
12    \fp_eval:n {round (\l_foo_tmpb_fp / 1cm)}y
13    =\fp_eval:n {round (\l_foo_tmpc_fp / (1cm * 1cm))}$
14  }
15  \ExplSyntaxOff
16  \draw (a) -- (b) node[midway,sloped,below] {\resultequation};
17  \ExplSyntaxOn
18  \path (1,-1) node[below,align=left] {
19    Actual ~ values:\\
20    $a = \fp_eval:n {\l_foo_tmpa_fp / 1cm}$\\
21    $b = \fp_eval:n {\l_foo_tmpb_fp / 1cm}$\\
22    $c = \fp_eval:n {\l_foo_tmpc_fp / (1cm * 1cm)}$
23  };
24  \ExplSyntaxOff
25 \end{tikzpicture}

```

`\trt_lt_get_intersection_line:NNNNNNNN`

`\trt_lt_get_intersection_line:NNNNNNNN<a1><b1><c1><a2><b2><c2><x><y>`

This function finds the intersection of lines $a_1x + b_1y = c_1$ and $a_2x + b_2y = c_2$, afterwards store the dimensions of the intersection in variables $\langle x \rangle$ and $\langle y \rangle$.

All arguments are floating points variables, $\langle x \rangle$ and $\langle y \rangle$ needs to be local variables.

`\trt_lt_get_intersection_coordinate:nnnnNN`

`\trt_lt_get_intersection_coordinate:nnnnNN
{<coor 1>}{<coor 2>}{<coor 3>}{<coor 4>}<x><y>`

This function is a wrapper of `\trt_lt_get_intersection_line:NNNNNNNN`. It finds the intersection of the line joining $\langle coor 1 \rangle$, $\langle coor 2 \rangle$ and the line joining $\langle coor 3 \rangle$, $\langle coor 4 \rangle$. The dimensions of the returned point is stored in $\langle x \rangle$ and $\langle y \rangle$, which are local l3fp variables.

A warning will be raised if the lines are parallel, in that case $\langle x \rangle$ and $\langle y \rangle$ are set to zero.

This is the base of intersection.

```

1 \begin{tikzpicture}
2   \draw (0,0) coordinate (a) -- (0,1) coordinate (b);
3   \draw (1,0) coordinate (c) -- (.5,1) coordinate (d);
4   \ExplSyntaxOn
5   \fp_new:N \l_foo_tmpa_fp
6   \fp_new:N \l_foo_tmpb_fp
7   \trt_lt_get_intersection_coordinate:nnnnNN {a} {b} {c} {d}
8     \l_foo_tmpa_fp \l_foo_tmpb_fp
9   \coordinate (i) at (\fp_to_dim:N \l_foo_tmpa_fp,
10     \fp_to_dim:N \l_foo_tmpb_fp);
11  \ExplSyntaxOff
12  \fill[maincolor] (i) circle (1.5pt);
13 \end{tikzpicture}

```

`\trt_lt_get_perpendicular_equation:nNNNNNN`

`\trt_lt_get_perpendicular_equation:nNNNNNN{<coor 1>}<a1><b1><c1><a2><b2><c2>`

This function finds the line of equation $a_2x + b_2y = c_2$ that passes coordinate $\langle coor 1 \rangle$ and is perpendicular to $a_1x + b_1y = c_1$.

(0.5, 4)



(3, 1)

(-1, 0)

Equation of perpendicular line:

$$4x + 1y = 6$$

Actual values:

$$a = 3.999998260240463$$

$$b = 0.9999995650601154$$

$$c = 5.999993708152788$$

```
1 \begin{tikzpicture}
2   \draw (-1,0) coordinate (b) node[left] {$(-1,0)$} --
3     (3,1) coordinate (c) node[right] {$ (3,1)$};
4   \fill (0.5,4) circle (1.5pt) coordinate (a) node[above] {$ (0.5,4)$};
5   \ExplSyntaxOn
6   \fp_new:N \l_foo_tmpa_fp
7   \fp_new:N \l_foo_tmpb_fp
8   \fp_new:N \l_foo_tmpe_fp
9   \fp_new:N \l_foo_tmpe_fp
10  \fp_new:N \l_foo_tmpe_fp
11  \fp_new:N \l_foo_tmpe_fp
12  \trt_lt_get_line_equation:nnNNN {b} {c}
13    \l_foo_tmpa_fp \l_foo_tmpe_fp \l_foo_tmpe_fp
14  \trt_lt_get_perpendicular_equation:nnNNNN {a}
15    \l_foo_tmpa_fp \l_foo_tmpe_fp \l_foo_tmpe_fp
16    \l_foo_tmpe_fp \l_foo_tmpe_fp \l_foo_tmpe_fp
17  \def \resultequation {
18    $\fp_eval:n {round (\l_foo_tmpe_fp / 1cm)}x +
19    \fp_eval:n {round (\l_foo_tmpe_fp / 1cm)}y
20    =\fp_eval:n {round (\l_foo_tmpe_fp / (1cm * 1cm))}$
21  }
22  \ExplSyntaxOff
23  \path (1,0) node[below=3mm,align=center]
24    {Equation of perpendicular line:\\\resultequation};
25  \ExplSyntaxOn
26  \path (1,-1.5) node[below,align=left] {
27    Actual ~ values:\\
28    $a = \fp_eval:n {\l_foo_tmpe_fp / 1cm}$\\
29    $b = \fp_eval:n {\l_foo_tmpe_fp / 1cm}$\\
30    $c = \fp_eval:n {\l_foo_tmpe_fp / (1cm * 1cm)}$
31  };
32  \ExplSyntaxOff
33 \end{tikzpicture}
```

`\trt_lt_get_perpendicular_coordinate:nnnNN`

`\trt_lt_get_perpendicular_coordinate:nnnNN{<coor 1>}{<coor 2>}{<coor 3>}{<x>}{<y>}`

Find the dimensions of the foot of perpendicular from $\langle coor\ 1 \rangle$ to the line joining $\langle coor\ 2 \rangle$ and $\langle coor\ 3 \rangle$. Afterwards store the dimensions found in $\langle x \rangle$ and $\langle y \rangle$.

This is the base of foot of perpendicular.

3.4.2 The barycentric coordinate system

`initialize_barycentric`

`/tikz/triangletools/initialize barycentric=(\langle coor 1 \rangle)(\langle coor 2 \rangle)(\langle coor 3 \rangle)`

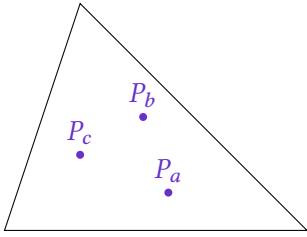
Use the three coordinates as “anchors” of the barycentric coordinate system.

`bc3`

`(bc3 cs: \langle l1 \rangle, \langle l2 \rangle, \langle l3 \rangle)`

Using the barycentric coordinate system. Note that the system needs to be initialized in advance using `initialize barycentric`, and an error message will be reported if you do otherwise.

The sum of $\langle l1 \rangle$, $\langle l2 \rangle$ and $\langle l3 \rangle$ is not necessarily 1 – the package will take care of that internally.



```

1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={initialize barycentric=(a)(b)(c)}] (bc3 cs:1,2,3)
6     circle[radius=1.5pt] node[above] {$P_a$};
7   \fill[maincolor,trt={initialize barycentric=(b)(c)(a)}] (bc3 cs:1,2,3)
8     circle[radius=1.5pt] node[above] {$P_b$};
9   \fill[maincolor,trt={initialize barycentric=(c)(a)(b)}] (bc3 cs:1,2,3)
10    circle[radius=1.5pt] node[above] {$P_c$};
11 \end{tikzpicture}

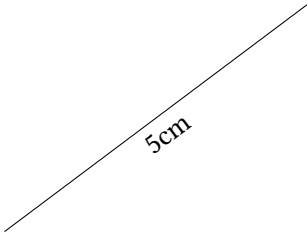
```

3.4.3 Distance-finding utility

`\trt_distance:nnN`

`\trt_distance:nnN {<coord 1>} {<coord 2>} <fp var>`

Find distance between `<coord 1>` and `<coord 2>`, and store that value to `<fp var>`.



```

1 \begin{tikzpicture}
2   \path (0,0) coordinate (a) (4,3) coordinate (b);
3   \ExplSyntaxOn
4   \fp_new:N \l_foo_tmpa_fp
5   \trt_distance:nnN {a} {b} \l_foo_tmpa_fp
6   \draw (a) -- (b) node[midway,sloped,below]
7     { \fp_eval:n {round(\l_foo_tmpa_fp / 1cm)} cm };
8   \ExplSyntaxOff
9 \end{tikzpicture}

```

`\trt_distance_triangle:nnnNNN`

`\trt_distance_triangle:nnnNNN{<coord 1>}{<coord 2>}{<coord 3>}<a><c>`

`\trt_distance:nnN` is needed to find the side lengths in a triangle (these side lengths are very helpful in many areas, for instance in this package to find special points based on the barycentric system). However, using it three times in a row is not quite elegant; this function is defined to automate that process.

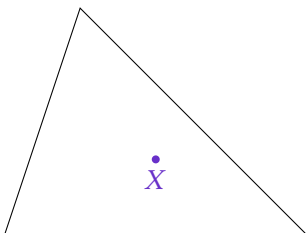
`<a>` is set to the distance between `<coord 2>` and `<coord 3>`, similar things happen for `` and `<c>`.

3.5 Customization

`output_name`

`/tikz/triangletools/output name=<name>`

This key can be used to change the name of the returned coordinates. The initial value of this key is `trt output`.



```

1 \begin{tikzpicture}[trt={output name=hello world}]
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={circumcenter=(a)(b)(c)}] (hello world)
6     circle[radius=1.5pt] node[below] {$X$};
7 \end{tikzpicture}

```

4 Implementation

1 `<@@=trt>`

4.1 The main package file

```
2 <*triangletools>
3 \RequirePackage{tikz}
4 \RequirePackage{expl3}
5 \ProvidesExplPackage {triangletools} {2020/04/30} {0.1}
6 {TikZ support for triangular geometry}
```

\trt@tmp@i We will use these dimensions many times to extract the dimensions of a TikZ coordinate.
\trt@tmp@ii

```
7 \newdimen\trt@tmp@i
8 \newdimen\trt@tmp@ii
```

(End definition for \trt@tmp@i and \trt@tmp@ii. These functions are documented on page ??.)

Let's load the necessary subpackage files.

```
9 \input {trtmessages.code.tex}
10 \input {trtlinetools.code.tex}
11 \input {trtbarycentric.code.tex}
12 \input {trtdistance.code.tex}
13 \input {trtspecialpoints.code.tex}
14 \input {trtfrontend.code.tex}
15 </triangletools>
```

4.2 Errors and warnings

```
16 <*messages>
17 \ProvidesExplFile {trtmessages.code.tex} {2020/04/30} {0.1}
18 {The ~ triangletools ~ package: ~ Messages}
```

We also need to declare some helpful messages that we will use later on.

In `trtlinetools.code.tex`, when we find the intersection of two lines, a warning will be shown if the lines are parallel. The warning is based on `intersection-not-found`.

```
19 \msg_new:nnnn {triangletools} {intersection-not-found}
20 {
21   Intersection ~ not ~ found.
22 }
23 {
24   You ~ told ~ me ~ to ~ find ~ the ~ intersection ~ of ~ the ~ line ~
25   joining ~ #1 ~ and ~ #2 ~ and ~ the ~ line ~ joining ~ #3 ~ and ~ #4, ~
26   however ~ these ~ lines ~ are ~ parallel ~ so ~ I ~ can't ~ find ~ any ~
27   intersection. ~ The ~ return ~ point ~ is ~ set ~ to ~ the ~ origin ~
28   (0, ~ 0).
29 }
```

When the barycentric coordinate system, implemented in `trtbarycentric.code.tex`, is used, it should already be initialized, *i.e.* we should already know what are the three “anchor” coordinates of the system. If the coordinate system is not yet initialized, this error will be shown.

```
30 \msg_new:nnnn {triangletools} {uninitialized}
31 {
32   Barycentric ~ coordinate ~ system ~ not ~ initialized.
33 }
34 {
35   You ~ have ~ not ~ initialized ~ the ~ three ~ anchor ~ points ~ for ~
36   the ~ coordinate ~ system. ~ Please ~ initialize ~ the ~ points ~
37   before ~ using ~ the ~ 'bc3' ~ coordinate ~ system.
38 }
```

We do let the user to find triangle center X_i for any i . However this package obviously can't implement all points in ETC (in fact, I will implement only some most

important points). An error will be raised if the user tries to use an unimplemented point.

```

39 \msg_new:nnnn {triangletools} {center-not-found}
40 {
41   Triangle ~ center ~ not ~ found.
42 }
43 {
44   I ~ can't ~ find ~ the ~ requested ~ triangle ~ center, ~ because ~
45   point ~ X(#1) ~ is ~ not ~ yet ~ implemented ~ in ~ the ~ triangletools ~
46   package. ~ Try ~ to ~ construct ~ it ~ yourself.
47 }

```

We need to guard against using `\trradius` before the macro stores something.

```

48 \msg_new:nnnn {triangletools} {no-radius-found}
49 {
50   No ~ circles ~ can ~ be ~ constructed.
51 }
52 {
53   I ~ can't ~ construct ~ the ~ requested ~ circle, ~ because ~ you ~ have ~
54   not ~ request ~ me ~ to ~ construct ~ any ~ triangle ~ centers ~ that ~
55   are ~ associated ~ to ~ a ~ circle. ~ I ~ will ~ set ~
56   \protect\trradius\space to ~ zero ~ now.
57 }
58 </messages>

```

4.3 The backend layer

4.3.1 The line tools utility

```

59 <*linetools>
60 \ProvidesExplFile {trtlinetools.code.tex} {2020/04/30} {0.1}
61 {The ~ triangletools ~ package: ~ Utilities ~ for ~ lines}

```

In `trtlinetools.code.tex`, we will implement the necessary functions to handle lines in a mathematical way.

Firstly, let's declare some internal variables that we will use later.

```

\__trt_lt_pointi_x_fp
\__trt_lt_pointi_y_fp
\__trt_lt_pointii_x_fp
\__trt_lt_pointii_y_fp
62 \fp_new:N \__trt_lt_pointi_x_fp
63 \fp_new:N \__trt_lt_pointi_y_fp
64 \fp_new:N \__trt_lt_pointii_x_fp
65 \fp_new:N \__trt_lt_pointii_y_fp

```

These variables are used to store coordinates of the two vertices of a segment.

(End definition for `__trt_lt_pointi_x_fp` and others.)

```

\__trt_lt_linei_a_fp
\__trt_lt_linei_b_fp
\__trt_lt_linei_c_fp
\__trt_lt_lineii_a_fp
\__trt_lt_lineii_b_fp
\__trt_lt_lineii_c_fp
66 \fp_new:N \__trt_lt_linei_a_fp
67 \fp_new:N \__trt_lt_linei_b_fp
68 \fp_new:N \__trt_lt_linei_c_fp
69 \fp_new:N \__trt_lt_lineii_a_fp
70 \fp_new:N \__trt_lt_lineii_b_fp
71 \fp_new:N \__trt_lt_lineii_c_fp

```

We will store the line equation under the format of $ax + by = c$, because this is the most generic format. These six variables will do that job.

(End definition for `__trt_lt_linei_a_fp` and others.)

```

\__trt_lt_tmp_fp
\__trt_lt_tmpa_fp
\__trt_lt_tmpb_fp
72 \fp_new:N \__trt_lt_tmp_fp
73 \fp_new:N \__trt_lt_tmpa_fp
74 \fp_new:N \__trt_lt_tmpb_fp

```

Some additional temporary variables.

(End definition for `\l__trt_lt_tmp_fp`, `\l__trt_lt_tmpa_fp`, and `\l__trt_lt_tmpb_fp`.)

`\trt_lt_get_line_equation:nnNNN` Find the equation of the line passing #1 and #2, and store the values of a , b , c found to #3, #4 and #5, which are floating point variables, respectively.

```

75 \cs_new:Npn \trt_lt_get_line_equation:nnNNN #1 #2 #3 #4 #5
76 {
77   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#1}{center}}
78   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#1}{center}}
79   \fp_set:Nn \l__trt_i_pointi_x_fp {\trt@tmp@i}
80   \fp_set:Nn \l__trt_i_pointi_y_fp {\trt@tmp@ii}
81   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#2}{center}}
82   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#2}{center}}
83   \fp_set:Nn \l__trt_i_pointii_x_fp {\trt@tmp@i}
84   \fp_set:Nn \l__trt_i_pointii_y_fp {\trt@tmp@ii}

```

There is a simple hack here. We have $ax_1 + by_1 = c = ax_2 + by_2$, which is equivalent to $a(x_1 - x_2) = b(y_2 - y_1)$. Therefore $a = y_2 - y_1$ and $b = x_1 - x_2$ can be used.

```

85   \fp_set:Nn #3
86   {
87     \l__trt_i_pointii_y_fp - \l__trt_i_pointi_y_fp
88   }
89   \fp_set:Nn #4
90   {
91     \l__trt_i_pointi_x_fp - \l__trt_i_pointii_x_fp
92   }
93   \fp_set:Nn #5
94   {
95     #3 * \l__trt_i_pointi_x_fp + #4 * \l__trt_i_pointi_y_fp
96   }
97 }

```

(End definition for `\trt_lt_get_line_equation:nnNNN`. This function is documented on page 6.)

`\trt_lt_get_intersection_line:NNNNNNNN` Find the intersection of two lines with given equation, after that store the intersection coordinate to floating point variables #7 and #8.

```

98 \cs_new:Npn \trt_lt_get_intersection_line:NNNNNNNN #1 #2 #3 #4 #5 #6 #7 #8
99 {
100   \fp_set:Nn \l__trt_lt_tmp_fp { #1 * #5 - #4 * #2 }

```

If `\l__trt_lt_tmp_fp` is zero, the two lines are parallel. In that case, we will issue a warning, and set the intersection coordinate to (0, 0). Otherwise, continue computing as usual.

```

101   \fp_compare:nNnTF {\l__trt_lt_tmp_fp} = {0}
102   {
103     \msg_warning:nnnnnn {triangletools} {intersection-not-found}
104     {(#1)} {(#2)} {(#3)} {(#4)}
105     \fp_set:Nn #7 {0}
106     \fp_set:Nn #8 {0}
107   }
108   {
109     \fp_set:Nn #7 { ( #5 * #3 - #2 * #6 ) / \l__trt_lt_tmp_fp }
110     \fp_set:Nn #8 { ( #1 * #6 - #4 * #3 ) / \l__trt_lt_tmp_fp }
111   }
112 }

```

(End definition for `\trt_lt_get_intersection_line:NNNNNNNN`. This function is documented on page 7.)

`\trt_lt_get_intersection_coordinate:nnnnNN` Let's generalize `\trt_lt_get_intersection_line:NNNNNNNN`. The following function finds the intersection of two lines between #1, #2 and #3, #4, and store the coordinates

to #5 and #6. We still use floating point variables here, as they might be useful in the future.

```

113 \cs_new:Npn \trt_lt_get_intersection_coordinate:nnnnNN #1 #2 #3 #4 #5 #6
114 {
115   \trt_lt_get_line_equation:nnNNN {#1} {#2}
116   \l__trt_lt_line_i_a_fp \l__trt_lt_line_i_b_fp \l__trt_lt_line_i_c_fp
117   \trt_lt_get_line_equation:nnNNN {#3} {#4}
118   \l__trt_lt_line_ii_a_fp \l__trt_lt_line_ii_b_fp \l__trt_lt_line_ii_c_fp
119   \trt_lt_get_intersection_line:NNNNNNNN
120   \l__trt_lt_line_i_a_fp \l__trt_lt_line_i_b_fp \l__trt_lt_line_i_c_fp
121   \l__trt_lt_line_ii_a_fp \l__trt_lt_line_ii_b_fp \l__trt_lt_line_ii_c_fp
122   #5 #6
123 }

```

(End definition for `\trt_lt_get_intersection_coordinate:nnnnNN`. This function is documented on page 7.)

`__trt_lt_return_intersection:nnnnn`

Now, let's TikZify the above function! Note that I use `overlay` because I don't want to affect the bounding box. The user can use the returned coordinate to change the bounding box in whatever way he wants to.

```

124 \cs_new:Npn \__trt_lt_return_intersection:nnnnn #1 #2 #3 #4 #5
125 {
126   \trt_lt_get_intersection_coordinate:nnnnNN {#1} {#2} {#3} {#4}
127   \l__trt_lt_tmpa_fp \l__trt_lt_tmpb_fp
128   \coordinate[overlay] (#5) at
129     (\fp_to_dim:N \l__trt_lt_tmpa_fp, \fp_to_dim:N \l__trt_lt_tmpb_fp);
130 }

```

(End definition for `__trt_lt_return_intersection:nnnnn`.)

Next, let's make some implementation regarding perpendicularity.

`\trt_lt_get_perpendicular_equation:nNNNNNN`

This function finds the equation of the line passing point and being perpendicular to a line having a given equation. The task is not quite complicated: note that lines $ax + by = c$ and $ay - bx = d$ are perpendicular.

```

131 \cs_new:Npn \trt_lt_get_perpendicular_equation:nNNNNNN #1 #2 #3 #4 #5 #6 #7
132 {
133   \fp_set:Nn #5 { -#3 }
134   \fp_set:Nn #6 { #2 }
135   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#1}{center}}
136   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#1}{center}}
137   \fp_set:Nn \l__trt_lt_tmpa_fp {\trt@tmp@i}
138   \fp_set:Nn \l__trt_lt_tmpb_fp {\trt@tmp@ii}
139   \fp_set:Nn #7 { #5 * \l__trt_lt_tmpa_fp + #6 * \l__trt_lt_tmpb_fp }
140 }

```

(End definition for `\trt_lt_get_perpendicular_equation:nNNNNNN`. This function is documented on page 7.)

`\trt_lt_get_perpendicular_coordinate:nnnNN`

The base implemented, let's find the foot of perpendicular from a point to a segment.

```

141 \cs_new:Npn \trt_lt_get_perpendicular_coordinate:nnnNN #1 #2 #3 #4 #5
142 {
143   \trt_lt_get_line_equation:nnNNN {#2} {#3}
144   \l__trt_lt_line_i_a_fp \l__trt_lt_line_i_b_fp \l__trt_lt_line_i_c_fp
145
146   \trt_lt_get_perpendicular_equation:nNNNNNN {#1}
147   \l__trt_lt_line_i_a_fp \l__trt_lt_line_i_b_fp \l__trt_lt_line_i_c_fp
148   \l__trt_lt_line_ii_a_fp \l__trt_lt_line_ii_b_fp \l__trt_lt_line_ii_c_fp
149
150   \trt_lt_get_intersection_line:NNNNNNNN
151   \l__trt_lt_line_i_a_fp \l__trt_lt_line_i_b_fp \l__trt_lt_line_i_c_fp

```

```

152 \l__trt_lt_lineii_a_fp \l__trt_lt_lineii_b_fp \l__trt_lt_lineii_c_fp
153 #4 #5
154 }

```

(End definition for `\trt_lt_get_perpendicular_coordinate:nnnNN`. This function is documented on page 8.)

`\trt_lt_return_perpendicular_coordinate:nnnn`

This is just a wrapper of `\trt_lt_get_perpendicular_coordinate:nnnNN`.

```

155 \cs_new:Npn \l__trt_lt_return_perpendicular_coordinate:nnnn #1 #2 #3 #4
156 {
157   \trt_lt_get_perpendicular_coordinate:nnnNN {#1} {#2} {#3}
158   \l__trt_lt_tmpa_fp \l__trt_lt_tmpb_fp
159   \coordinate[overlay] (#4) at
160   (\fp_to_dim:N \l__trt_lt_tmpa_fp, \fp_to_dim:N \l__trt_lt_tmpb_fp);
161 }
162 </linetools>

```

(End definition for `\l__trt_lt_return_perpendicular_coordinate:nnnn`.)

4.3.2 The barycentric coordinate system utility

```

163 <*barycentric>
164 \ProvidesExplFile {trtbarycentric.code.tex} {2020/04/30} {0.1}
165 {
166   The ~ triangletools ~ package:~ Utilities ~ for ~ the ~ barycentric ~
167   coordinate ~ system.
168 }

```

In `trtbarycentric.code.tex`, we will implement the three-point barycentric coordinate system, which is essential in constructing many special points in a triangle.

```

\l__trt_bc_anchor_ix_fp
\l__trt_bc_anchor_iy_fp
\l__trt_bc_anchor_iix_fp
\l__trt_bc_anchor_iiy_fp
\l__trt_bc_anchor_iiix_fp
\l__trt_bc_anchor_iiiy_fp

```

We use six variables to store the ‘anchors’ of the barycentric coordinate system.

```

169 \fp_new:N \l__trt_bc_anchor_ix_fp
170 \fp_new:N \l__trt_bc_anchor_iy_fp
171 \fp_new:N \l__trt_bc_anchor_iix_fp
172 \fp_new:N \l__trt_bc_anchor_iiy_fp
173 \fp_new:N \l__trt_bc_anchor_iiix_fp
174 \fp_new:N \l__trt_bc_anchor_iiiy_fp

```

(End definition for `\l__trt_bc_anchor_ix_fp` and others.)

```

\l__trt_bc_lambda_i_fp
\l__trt_bc_lambda_ii_fp
\l__trt_bc_lambda_iii_fp

```

We use these variables to store the user input coordinate. Note that our system is a three-point one, hence exactly three number is required.

Why lambda λ ? Well, I don’t know. Wikipedia uses that, so I do the same.

```

175 \fp_new:N \l__trt_bc_lambda_i_fp
176 \fp_new:N \l__trt_bc_lambda_ii_fp
177 \fp_new:N \l__trt_bc_lambda_iii_fp

```

(End definition for `\l__trt_bc_lambda_i_fp`, `\l__trt_bc_lambda_ii_fp`, and `\l__trt_bc_lambda_iii_fp`.)

`\l__trt_bc_initialized_bool`

We need to guard against using the system before initializing. This boolean variable does that job: if it is set to false (default), do nothing.

```

178 \bool_new:N \l__trt_bc_initialized_bool

```

(End definition for `\l__trt_bc_initialized_bool`.)

`\l__trt_bc_tmp_fp`

A temporary variable.

```

179 \fp_new:N \l__trt_bc_tmp_fp

```

(End definition for \l__trt_bc_tmp_fp.)

__trt_bc_initialize:nnn Initialize the barycentric coordinate system. This is the only place where \l__trt_bc_initialized_bool can be set to true, so this function must be executed before everything else in this file.

```

180 \cs_new:Npn \__trt_bc_initialize:nnn #1 #2 #3
181 {
182   \bool_set_true:N \__trt_bc_initialized_bool
183   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#1}{center}}
184   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#1}{center}}
185   \fp_set:Nn \__trt_bc_anchor_ix_fp {\trt@tmp@i}
186   \fp_set:Nn \__trt_bc_anchor_iy_fp {\trt@tmp@ii}
187   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#2}{center}}
188   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#2}{center}}
189   \fp_set:Nn \__trt_bc_anchor_iix_fp {\trt@tmp@i}
190   \fp_set:Nn \__trt_bc_anchor_iiy_fp {\trt@tmp@ii}
191   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#3}{center}}
192   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#3}{center}}
193   \fp_set:Nn \__trt_bc_anchor_iiix_fp {\trt@tmp@i}
194   \fp_set:Nn \__trt_bc_anchor_iiiy_fp {\trt@tmp@ii}
195 }

```

(End definition for __trt_bc_initialize:nnn.)

bc3 The bc3 coordinate system implementation. We will guard against using it when __trt_bc_initialize:nnn is not yet executed – in that case, uninitialized error will be raised.

We will receive arguments of bc3 as #1,#2,#3, so a simple parser is needed. All interesting things will be done with that parser.

```

196 \tikzdeclarecoordinatesystem {bc3}
197 {
198   \bool_if:NTF \__trt_bc_initialized_bool
199   {
200     \__trt_bc_parse:w #1 \q_stop
201   }
202   {
203     \msg_error:nn {triangletools} {uninitialized}
204   }
205 }

```

(End definition for bc3. This function is documented on page 8.)

__trt_bc_parse:w This is the parser we use for bc3.

The conversion from λ_i to the Cartesian format is pretty simple, we have $x = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$ and the formula for y is similar. However, first we have to change the value of λ_i so that $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

```

206 \cs_new:Npn \__trt_bc_parse:w #1,#2,#3 \q_stop
207 {
208   \fp_set:Nn \__trt_bc_tmp_fp { (#1) + (#2) + (#3) }
209   \fp_set:Nn \__trt_bc_lambda_i_fp { (#1) / (\__trt_bc_tmp_fp) }
210   \fp_set:Nn \__trt_bc_lambda_ii_fp { (#2) / (\__trt_bc_tmp_fp) }
211   \fp_set:Nn \__trt_bc_lambda_iii_fp { (#3) / (\__trt_bc_tmp_fp) }
212   \fp_set:Nn \__trt_tmp_a_fp
213   {
214     \__trt_bc_anchor_ix_fp * \__trt_bc_lambda_i_fp +
215     \__trt_bc_anchor_iix_fp * \__trt_bc_lambda_ii_fp +
216     \__trt_bc_anchor_iiix_fp * \__trt_bc_lambda_iii_fp
217   }
218   \fp_set:Nn \__trt_tmp_b_fp

```

```

219     {
220       \l__trt_bc_anchor_iy_fp * \l__trt_bc_lambda_i_fp +
221       \l__trt_bc_anchor_iiy_fp * \l__trt_bc_lambda_ii_fp +
222       \l__trt_bc_anchor_iiiy_fp * \l__trt_bc_lambda_iiiy_fp
223     }

```

Floating point variables are not \TeX dimensions, hence `\fp_to_dim:N` is used.

```

224     \pgf@x = \fp_to_dim:N \l__trt_tmp_a_fp
225     \pgf@y = \fp_to_dim:N \l__trt_tmp_b_fp
226   }
227 </barycentric>

```

(End definition for `__trt_bc_parse:w`.)

4.3.3 Distance-finding utility

```

228 <*distance>
229 \ProvidesExplFile {trtdistance.code.tex} {2020/04/30} {0.1}
230 {The ~ triangletools ~ package: ~ Utilities ~ for ~ 2d ~ distance}

```

This file implements functions to find the distance between (2d) TikZ coordinates.

These variables are used to store the coordinates of the points between which we are finding the distance.

```

\l__trt_d_pointi_x_fp
\l__trt_d_pointi_y_fp
\l__trt_d_pointii_x_fp
\l__trt_d_pointii_y_fp
231 \fp_new:N \l__trt_d_pointi_x_fp
232 \fp_new:N \l__trt_d_pointii_x_fp
233 \fp_new:N \l__trt_d_pointi_y_fp
234 \fp_new:N \l__trt_d_pointii_y_fp

```

(End definition for `\l__trt_d_pointi_x_fp` and others.)

`\trt_distance:nnN` Find the distance between TikZ coordinates #1 and #2.

```

235 \cs_new:Npn \trt_distance:nnN #1 #2 #3
236 {
237   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#1}{center}}
238   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#1}{center}}
239   \fp_set:Nn \l__trt_d_pointi_x_fp {\trt@tmp@i}
240   \fp_set:Nn \l__trt_d_pointi_y_fp {\trt@tmp@ii}
241   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#2}{center}}
242   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#2}{center}}
243   \fp_set:Nn \l__trt_d_pointii_x_fp {\trt@tmp@i}
244   \fp_set:Nn \l__trt_d_pointii_y_fp {\trt@tmp@ii}
245   \fp_set:Nn #3
246   {
247     sqrt((
248       (\l__trt_d_pointi_x_fp - \l__trt_d_pointii_x_fp) *
249       (\l__trt_d_pointi_x_fp - \l__trt_d_pointii_x_fp)
250     ) + (
251       (\l__trt_d_pointi_y_fp - \l__trt_d_pointii_y_fp) *
252       (\l__trt_d_pointi_y_fp - \l__trt_d_pointii_y_fp)
253     ))
254   }
255 }

```

(End definition for `\trt_distance:nnN`. This function is documented on page 9.)

`\trt_distance_triangle:nnnNNN` We mainly need the above function to find the side length in a triangle. Let's create a function that do so automatically.

```

256 \cs_new:Npn \trt_distance_triangle:nnnNNN #1 #2 #3 #4 #5 #6
257 {
258   \trt_distance:nnN {#2} {#3} #4

```



```

259 \trt_distance:nnN {#3} {#1} #5
260 \trt_distance:nnN {#1} {#2} #6
261 }
262 </distance>

```

(End definition for `\trt_distance_triangle:nnnNNN`. This function is documented on page 9.)

4.4 Construction of triangle centers

```

263 <*specialpoints>
264 \ProvidesExplFile {trtspecialpoints.code.tex} {2020/04/30} {0.1}
265 {The ~ triangletools ~ package: ~ Triangle ~ center ~ construction}

```

This file will use the utility implemented in the above sections to find some most important triangle centers described in the ETC.

`\l__trt_sp_a_fp` We will need the side length of the triangle for some centers.

```

\l__trt_sp_b_fp
\l__trt_sp_c_fp
266 \fp_new:N \l__trt_sp_a_fp
267 \fp_new:N \l__trt_sp_b_fp
268 \fp_new:N \l__trt_sp_c_fp

```

(End definition for `\l__trt_sp_a_fp`, `\l__trt_sp_b_fp`, and `\l__trt_sp_c_fp`.)

`\l__trt_sp_coordinatei_x_fp` These variables may also be helpful for triangle centers for which a simple formula doesn't exist, e.g. the circumcenter.

```

\l__trt_sp_coordinatei_y_fp
\l__trt_sp_coordinateii_x_fp
\l__trt_sp_coordinateii_y_fp
\l__trt_sp_coordinateiii_x_fp
\l__trt_sp_coordinateiii_y_fp
269 \fp_new:N \l__trt_sp_coordinatei_x_fp
270 \fp_new:N \l__trt_sp_coordinatei_y_fp
271 \fp_new:N \l__trt_sp_coordinateii_x_fp
272 \fp_new:N \l__trt_sp_coordinateii_y_fp
273 \fp_new:N \l__trt_sp_coordinateiii_x_fp
274 \fp_new:N \l__trt_sp_coordinateiii_y_fp
275 \fp_new:N \l__trt_sp_linei_a_fp
276 \fp_new:N \l__trt_sp_linei_b_fp
277 \fp_new:N \l__trt_sp_linei_c_fp
278 \fp_new:N \l__trt_sp_lineii_a_fp
279 \fp_new:N \l__trt_sp_lineii_b_fp
280 \fp_new:N \l__trt_sp_lineii_c_fp

```

(End definition for `\l__trt_sp_coordinatei_x_fp` and others.)

`\l__trt_sp_tmpa_fp` Some additional temporary variables.

```

\l__trt_sp_tmppb_fp
\l__trt_sp_tmppc_fp
281 \fp_new:N \l__trt_sp_tmpa_fp
282 \fp_new:N \l__trt_sp_tmppb_fp
283 \fp_new:N \l__trt_sp_tmppc_fp

```

(End definition for `\l__trt_sp_tmpa_fp`, `\l__trt_sp_tmppb_fp`, and `\l__trt_sp_tmppc_fp`.)

4.4.1 X_1 – The incenter

Each center will have a function taking four arguments. The first three arguments are the TikZ coordinates of the triangle vertices; the last argument is the name of the return TikZ coordinate.

To prevent conflict between these sister functions when they are used together, I put each of them inside a \TeX group.

`\trt_sp_incenter:nnnn` Return the incenter. It is based on the barycentric coordinate of the incenter, (a, b, c) .

```

284 \cs_new:Npn \trt_sp_incenter:nnnn #1 #2 #3 #4
285 {

```

```

286 \group_begin:
287 \__trt_bc_initialize:nnn {#1} {#2} {#3}
288 \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
289 \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
290 \path[overlay] (bc3 ~ cs \c_colon_str
291 \fp_eval:n {\l__trt_sp_a_fp},
292 \fp_eval:n {\l__trt_sp_b_fp},
293 \fp_eval:n {\l__trt_sp_c_fp}) coordinate (#4);
294 \group_end:
295 }

```

(End definition for `\trt_sp_incenter:nnnn`. This function is documented on page 3.)

`\trt_sp_excenter:nnnn` Return the excenter of the triangle, with respect to vertex #1. This center is just a derivation of the incenter; also it is not unique, so it is not assigned a number. Barycentric coordinate of the excenter is $(-a, b, c)$, where a is the length of the side joining #2 and #3.

Note that this is the only function in this series in which argument order is important.

```

296 \cs_new:Npn \trt_sp_excenter:nnnn #1 #2 #3 #4
297 {
298 \group_begin:
299 \__trt_bc_initialize:nnn {#1} {#2} {#3}
300 \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
301 \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
302 \path[overlay] (bc3 ~ cs \c_colon_str
303 \fp_eval:n {- \l__trt_sp_a_fp},
304 \fp_eval:n {\l__trt_sp_b_fp},
305 \fp_eval:n {\l__trt_sp_c_fp}) coordinate (#4);
306 \group_end:
307 }

```

(End definition for `\trt_sp_excenter:nnnn`. This function is documented on page 3.)

4.4.2 X_2 – The centroid

`\trt_sp_centroid:nnnn` This is perhaps the simplest of all. Barycentric coordinate: $(1, 1, 1)$.

```

308 \cs_new:Npn \trt_sp_centroid:nnnn #1 #2 #3 #4
309 {
310 \group_begin:
311 \__trt_bc_initialize:nnn {#1} {#2} {#3}
312 \path[overlay] (bc3 ~ cs \c_colon_str 1, 1, 1) coordinate (#4);
313 \group_end:
314 }

```

(End definition for `\trt_sp_centroid:nnnn`. This function is documented on page 3.)

4.4.3 X_3 – The circumcenter

`\trt_sp_circumcenter:nnnn` This is opposite to X_2 : perhaps this is the most complex of all. The barycentric coordinate formula is not simple enough for me, so I construct this point purely manually: find the intersection of the perpendicular bisectors.

```

315 \cs_new:Npn \trt_sp_circumcenter:nnnn #1 #2 #3 #4
316 {
317 \group_begin:

```

Firstly, let's store the coordinate of the vertices.

```

318 \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#1}{center}}

```

```

319 \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#1}{center}}
320 \fp_set:Nn \l__trt_sp_coordinatei_x_fp {\trt@tmp@i}
321 \fp_set:Nn \l__trt_sp_coordinatei_y_fp {\trt@tmp@ii}
322 \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#2}{center}}
323 \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#2}{center}}
324 \fp_set:Nn \l__trt_sp_coordinateii_x_fp {\trt@tmp@i}
325 \fp_set:Nn \l__trt_sp_coordinateii_y_fp {\trt@tmp@ii}
326 \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#3}{center}}
327 \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#3}{center}}
328 \fp_set:Nn \l__trt_sp_coordinateiii_x_fp {\trt@tmp@i}
329 \fp_set:Nn \l__trt_sp_coordinateiii_y_fp {\trt@tmp@ii}

```

Now, let's change point #2 to the midpoint between #1 and #2, and do the same for #3.

```

330 \fp_set:Nn \l__trt_sp_coordinateii_x_fp
331 {
332   (\l__trt_sp_coordinatei_x_fp + \l__trt_sp_coordinateii_x_fp) / 2
333 }
334 \fp_set:Nn \l__trt_sp_coordinateii_y_fp
335 {
336   (\l__trt_sp_coordinatei_y_fp + \l__trt_sp_coordinateii_y_fp) / 2
337 }
338 \coordinate[overlay] (trt@tmp@ii) at (
339   \fp_to_dim:N \l__trt_sp_coordinateii_x_fp,
340   \fp_to_dim:N \l__trt_sp_coordinateii_y_fp);
341 \fp_set:Nn \l__trt_sp_coordinateiii_x_fp
342 {
343   (\l__trt_sp_coordinatei_x_fp + \l__trt_sp_coordinateiii_x_fp) / 2
344 }
345 \fp_set:Nn \l__trt_sp_coordinateiii_y_fp
346 {
347   (\l__trt_sp_coordinatei_y_fp + \l__trt_sp_coordinateiii_y_fp) / 2
348 }
349 \coordinate[overlay] (trt@tmp@iii) at (
350   \fp_to_dim:N \l__trt_sp_coordinateiii_x_fp,
351   \fp_to_dim:N \l__trt_sp_coordinateiii_y_fp);

```

All we have to do now is to find the equations of the bisectors and their intersection.

```

352 \trt_lt_get_line_equation:nnNNN {#1} {#2}
353   \l__trt_sp_tmpa_fp \l__trt_sp_tmpb_fp \l__trt_sp_tmpc_fp
354 \trt_lt_get_perpendicular_equation:nNNNNNN {trt@tmp@ii}
355   \l__trt_sp_tmpa_fp \l__trt_sp_tmpb_fp \l__trt_sp_tmpc_fp
356   \l__trt_sp_linei_a_fp \l__trt_sp_linei_b_fp \l__trt_sp_linei_c_fp
357 \trt_lt_get_line_equation:nnNNN {#1} {#3}
358   \l__trt_sp_tmpa_fp \l__trt_sp_tmpb_fp \l__trt_sp_tmpc_fp
359 \trt_lt_get_perpendicular_equation:nNNNNNN {trt@tmp@iii}
360   \l__trt_sp_tmpa_fp \l__trt_sp_tmpb_fp \l__trt_sp_tmpc_fp
361   \l__trt_sp_lineii_a_fp \l__trt_sp_lineii_b_fp \l__trt_sp_lineii_c_fp
362 \trt_lt_get_intersection_line:NNNNNNNN
363   \l__trt_sp_linei_a_fp \l__trt_sp_linei_b_fp \l__trt_sp_linei_c_fp
364   \l__trt_sp_lineii_a_fp \l__trt_sp_lineii_b_fp \l__trt_sp_lineii_c_fp
365   \l__trt_sp_tmpa_fp \l__trt_sp_tmpb_fp
366 \coordinate[overlay] (#4) at (
367   \fp_to_dim:N \l__trt_sp_tmpa_fp, \fp_to_dim:N \l__trt_sp_tmpb_fp);
368 \group_end:
369 }

```

Quite surprisingly, the function is still very fast after all this. On my machine it never exceeds 10ms in execution time.

(End definition for `\trt_sp_circumcenter:nnnn`. This function is documented on page 3.)

4.4.4 X_4 – The orthocenter

`\trt_sp_orthocenter:nnnn` Return the orthocenter of the triangle. This point is also constructed manually instead of using a proved formula. However, the utilities help making the construction look very simple.

```

370 \cs_new:Npn \trt_sp_orthocenter:nnnn #1 #2 #3 #4
371 {
372   \group_begin:
373     \__trt_lt_return_perpendicular_coordinate:nnnn {#1} {#2} {#3} {trt@tmp@i}
374     \__trt_lt_return_perpendicular_coordinate:nnnn {#2} {#1} {#3} {trt@tmp@ii}
375     \__trt_lt_return_intersection:nnnnn
376       {#1} {trt@tmp@i} {#2} {trt@tmp@ii} {#4}
377   \group_end:
378 }
```

(End definition for `\trt_sp_orthocenter:nnnn`. This function is documented on page 4.)

4.4.5 X_5 – The nine-point center

`\trt_sp_ninepointcenter:nnnn` Return the center of the nine-point circle.

```

379 \cs_new:Npn \trt_sp_ninepointcenter:nnnn #1 #2 #3 #4
380 {
381   \group_begin:
```

X_5 is the midpoint of X_3 and X_4 . Therefore, for simplicity, X_3 and X_4 are constructed first. This causes some run-time overhead, however the overall execution time is still below 15ms, which is, in my opinion, still good.

Note that we already used `trt@tmp@i` and `trt@tmp@ii` coordinates in the construction of X_3 and X_4 , so to prevent conflict, `trt@tmp@iii` and `trt@tmp@iv` are used.

```

382   \trt_sp_circumcenter:nnnn {#1} {#2} {#3} {trt@tmp@iii}
383   \trt_sp_orthocenter:nnnn {#1} {#2} {#3} {trt@tmp@iv}
384   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{trt@tmp@iii}{center}}
385   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{trt@tmp@iii}{center}}
386   \fp_set:Nn \__trt_sp_tmpa_fp {\trt@tmp@i}
387   \fp_set:Nn \__trt_sp_tmpb_fp {\trt@tmp@ii}
388   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{trt@tmp@iv}{center}}
389   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{trt@tmp@iv}{center}}
390   \fp_set:Nn \__trt_sp_tmpa_fp { (\trt@tmp@i + \__trt_sp_tmpa_fp) / 2 }
391   \fp_set:Nn \__trt_sp_tmpb_fp { (\trt@tmp@ii + \__trt_sp_tmpb_fp) / 2 }
392   \coordinate[overlay] (#4) at (\fp_to_dim:N \__trt_sp_tmpa_fp,
393     \fp_to_dim:N \__trt_sp_tmpb_fp);
394   \group_end:
395 }
```

(End definition for `\trt_sp_ninepointcenter:nnnn`. This function is documented on page 4.)

4.4.6 X_6 – The symmedian point

`\trt_sp_symmedian:nnnn` Return the symmedian point (*aka.* the Lemoine point or Grebe point). The barycentric coordinate of the point is (a^2, b^2, c^2) .

```

396 \cs_new:Npn \trt_sp_symmedian:nnnn #1 #2 #3 #4
397 {
398   \group_begin:
399     \__trt_bc_initialize:nnn {#1} {#2} {#3}
400     \trt_distance_triangle:nnNNN {#1} {#2} {#3}
401     \__trt_sp_a_fp \__trt_sp_b_fp \__trt_sp_c_fp
402     \path[overlay] (bc3 ~ cs \c_colon_str
403       \fp_eval:n {\__trt_sp_a_fp * \__trt_sp_a_fp},
404       \fp_eval:n {\__trt_sp_b_fp * \__trt_sp_b_fp},
```

```

405         \fp_eval:n {\l__trt_sp_c_fp * \l__trt_sp_c_fp} coordinate (#4);
406     \group_end:
407 }

```

(End definition for `\trt_sp_symmedian:nnnn`. This function is documented on page 4.)

4.4.7 X_7 – The Gergonne point

`\trt_sp_gergonne:nnnn` Return the Gergonne point of the triangle. The barycentric coordinate of the point is $(\frac{1}{b+c-a}, \frac{1}{c+a-b}, \frac{1}{a+b-c})$.

```

408 \cs_new:Npn \trt_sp_gergonne:nnnn #1 #2 #3 #4
409 {
410     \group_begin:
411     \__trt_bc_initialize:nnn {#1} {#2} {#3}
412     \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
413     \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
414     \path[overlay] (bc3 ~ cs \c_colon_str
415         \fp_eval:n { 1/(\l__trt_sp_b_fp + \l__trt_sp_c_fp - \l__trt_sp_a_fp) },
416         \fp_eval:n { 1/(\l__trt_sp_c_fp + \l__trt_sp_a_fp - \l__trt_sp_b_fp) },
417         \fp_eval:n { 1/(\l__trt_sp_a_fp + \l__trt_sp_b_fp - \l__trt_sp_c_fp) }
418     ) coordinate (#4);
419     \group_end:
420 }

```

(End definition for `\trt_sp_gergonne:nnnn`. This function is documented on page 4.)

4.4.8 X_8 – The Nagel point

`\trt_sp_nagel:nnnn` Return the Nagel point. The barycentric coordinate of the point is $(b+c-a, c+a-b, a+b-c)$.

```

421 \cs_new:Npn \trt_sp_nagel:nnnn #1 #2 #3 #4
422 {
423     \group_begin:
424     \__trt_bc_initialize:nnn {#1} {#2} {#3}
425     \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
426     \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
427     \path[overlay] (bc3 ~ cs \c_colon_str
428         \fp_eval:n { \l__trt_sp_b_fp + \l__trt_sp_c_fp - \l__trt_sp_a_fp },
429         \fp_eval:n { \l__trt_sp_c_fp + \l__trt_sp_a_fp - \l__trt_sp_b_fp },
430         \fp_eval:n { \l__trt_sp_a_fp + \l__trt_sp_b_fp - \l__trt_sp_c_fp }
431     ) coordinate (#4);
432     \group_end:
433 }

```

(End definition for `\trt_sp_nagel:nnnn`. This function is documented on page 5.)

4.4.9 X_9 – The *mittenpunkt*

`\trt_sp_mittenpunkt:nnnn` Return the *mittenpunkt* of the triangle – its barycentric coordinate is $(a \times (b+c-a), b \times (c+a-b), c \times (a+b-c))$.

```

434 \cs_new:Npn \trt_sp_mittenpunkt:nnnn #1 #2 #3 #4
435 {
436     \group_begin:
437     \__trt_bc_initialize:nnn {#1} {#2} {#3}
438     \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
439     \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
440     \path[overlay] (bc3 ~ cs \c_colon_str
441         \fp_eval:n
442         {

```

```

443         \l__trt_sp_a_fp * (
444             \l__trt_sp_b_fp + \l__trt_sp_c_fp - \l__trt_sp_a_fp
445         )
446     },
447     \fp_eval:n
448     {
449         \l__trt_sp_b_fp * (
450             \l__trt_sp_c_fp + \l__trt_sp_a_fp - \l__trt_sp_b_fp
451         )
452     },
453     \fp_eval:n
454     {
455         \l__trt_sp_c_fp * (
456             \l__trt_sp_a_fp + \l__trt_sp_b_fp - \l__trt_sp_c_fp
457         )
458     }
459 ) coordinate (#4);
460 \group_end:
461 }

```

(End definition for `\trt_sp_mittenpunkt:nnnn`. This function is documented on page 5.)

4.4.10 X_{10} – The Spieker point

`\trt_sp_spieker:nnnn` Return the Spieker point. The barycentric coordinate of the point is $(b+c, c+a, a+b)$.

```

462 \cs_new:Npn \trt_sp_spieker:nnnn #1 #2 #3 #4
463 {
464     \group_begin:
465     \__trt_bc_initialize:nnn {#1} {#2} {#3}
466     \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
467     \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
468     \path[overlay] (bc3 ~ cs \c_colon_str
469         \fp_eval:n { \l__trt_sp_b_fp + \l__trt_sp_c_fp },
470         \fp_eval:n { \l__trt_sp_c_fp + \l__trt_sp_a_fp },
471         \fp_eval:n { \l__trt_sp_a_fp + \l__trt_sp_b_fp }
472     ) coordinate (#4);
473     \group_end:
474 }
475 </specialpoints>

```

(End definition for `\trt_sp_spieker:nnnn`. This function is documented on page 5.)

4.5 The frontend layer

```

476 <frontend>
477 \ProvidesExplFile {trtfrontend.code.tex} {2020/04/30} {0.1}
478 {The ~ triangletools ~ package: ~ The ~ front-end ~ layer}

```

The user interface of the package, which consists solely of pgf keys, will be implemented in this file.

`\l__trt_fr_output_name_tl` Store the name of the output coordinate. Default to `trt_output`.

```

479 \tl_new:N \l__trt_fr_output_name_tl
480 \tl_set:Nn \l__trt_fr_output_name_tl {trt ~ output}

```

(End definition for `\l__trt_fr_output_name_tl`.)

`\l__trt_fr_center_number_int` We only provide specific key for X_1, X_2, X_3 and X_4 . All other points can be referenced using a single generic key. We need to store the index of that point so that we can choose the right function for the point.

```

481 \int_new:N \l__trt_fr_center_number_int

```

(End definition for \l__trt_fr_center_number_int.)

\trtradius This macro will store the radius if a circle is associated. Of course firstly we need a floating point variable specified for that purpose.

\l__trt_fr_radius_fp

```
482 \fp_new:N \l__trt_fr_radius_fp
483 \cs_gset_nopar:Npn \trtradius
484 {
485   \msg_error:nn {triangletools} {no-radius-found} 0pt
486 }
```

(End definition for \trtradius and \l__trt_fr_radius_fp. This function is documented on page 2.)

trt Now it's time for the keys. They will be stored under /tikz/triangletools and can be accessed at trt={⟨keys⟩}.

```
487 \tikzset {
488   triangletools/.is ~ family,
489   trt/.code={\pgfkeys{/tikz/triangletools/.cd,#1}},
490   triangletools/.cd,
```

(End definition for trt. This function is documented on page 2.)

output_name Change the output name of all returned coordinates.

```
491 output ~ name/.code={
492   \tl_set:Nn \l__trt_fr_output_name_tl {#1}
493 },
```

(End definition for output name. This function is documented on page 9.)

intersection The front-end of the line tools utility.
foot_of_perpendicular

```
494 intersection/.code ~ args={(#1)(#2)--(#3)(#4)}{
495   \__trt_lt_return_intersection:nnnn {#1} {#2} {#3} {#4}
496   {\tl_use:N \l__trt_fr_output_name_tl}
497 },
498 foot ~ of ~ perpendicular/.code ~ args={(#1)--(#2)(#3)}{
499   \__trt_lt_return_perpendicular_coordinate:nnnn {#1} {#2} {#3}
500   {\tl_use:N \l__trt_fr_output_name_tl}
501 },
```

(End definition for intersection and foot of perpendicular. These functions are documented on page 6.)

initialize_barycentric The front-end of the barycentric coordinate system.

```
502 initialize ~ barycentric/.code ~ args={(#1)(#2)(#3)}{
503   \__trt_bc_initialize:nnn {#1} {#2} {#3}
504 },
```

(End definition for initialize barycentric. This function is documented on page 8.)

incenter The front-end of the triangle centers X_1 to X_4 and the excenter.
excenter
centroid
circumcenter
orthocenter

```
505 incenter/.code ~ args={(#1)(#2)(#3)}{
506   \trt_sp_incenter:nnnn {#1} {#2} {#3} {trt@tmp@center}
507   \pgfkeysalso{foot ~ of ~ perpendicular=(trt@tmp@center)--(#1)(#2)}
508   \trt_distance:nnN {\tl_use:N \l__trt_fr_output_name_tl} {trt@tmp@center}
509   \l__trt_fr_radius_fp
510   \cs_gset_nopar:Npx \trtradius { \fp_to_dim:N \l__trt_fr_radius_fp }
511   \coordinate (\tl_use:N \l__trt_fr_output_name_tl) at (trt@tmp@center);
512 },
513 excenter/.code ~ args={(#1)(#2)(#3)}{
```

```

514 \trt_sp_excenter:nnnn {#1} {#2} {#3} {trt@tmp@center}
515 \pgfkeysalso{foot ~ of ~ perpendicular=(trt@tmp@center)--(#2)(#3)}
516 \trt_distance:nnN {\tl_use:N \l__trt_fr_output_name_tl} {trt@tmp@center}
517 \l__trt_fr_radius_fp
518 \cs_gset_nopar:Npx \trradius { \fp_to_dim:N \l__trt_fr_radius_fp }
519 \coordinate (\tl_use:N \l__trt_fr_output_name_tl) at (trt@tmp@center);
520 },
521 centroid/.code ~ args={(#1)(#2)(#3)}{
522 \trt_sp_centroid:nnnn {#1} {#2} {#3}
523 {\tl_use:N \l__trt_fr_output_name_tl}
524 },
525 circumcenter/.code ~ args={(#1)(#2)(#3)}{
526 \trt_sp_circumcenter:nnnn {#1} {#2} {#3}
527 {\tl_use:N \l__trt_fr_output_name_tl}
528 \trt_distance:nnN {\tl_use:N \l__trt_fr_output_name_tl} {#1}
529 \l__trt_fr_radius_fp
530 \cs_gset_nopar:Npx \trradius { \fp_to_dim:N \l__trt_fr_radius_fp }
531 },
532 orthocenter/.code ~ args={(#1)(#2)(#3)}{
533 \trt_sp_orthocenter:nnnn {#1} {#2} {#3}
534 {\tl_use:N \l__trt_fr_output_name_tl}
535 },

```

(End definition for *incenter* and others. These functions are documented on page 3.)

triangle_center This key is used to access all centers. I don't give any centers from X_5 a key – this key is necessary to construct them.

```

536 triangle ~ center/.code ~ args={(#1)(#2)(#3)(#4)}{
537 \int_case:nnF {#4}
538 {
539 {1} {
540 \pgfkeysalso{incenter=(#1)(#2)(#3)}
541 }
542 {2} {
543 \pgfkeysalso{centroid=(#1)(#2)(#3)}
544 }
545 {3} {
546 \pgfkeysalso{circumcenter=(#1)(#2)(#3)}
547 }
548 {4} {
549 \pgfkeysalso{orthocenter=(#1)(#2)(#3)}
550 }
551 {5} {
552 \trt_sp_ninepointcenter:nnnn {#1} {#2} {#3}
553 {\tl_use:N \l__trt_fr_output_name_tl}
554 \group_begin:
555 \l__trt_bc_initialize:nnn {#1} {#2} {#3}
556 \coordinate (trt@tmp@mid) at (bc3 ~ cs \c_colon_str 1,1,0);
557 \group_end:
558 \trt_distance:nnN {\tl_use:N \l__trt_fr_output_name_tl} {trt@tmp@mid}
559 \l__trt_fr_radius_fp
560 \cs_gset_nopar:Npx \trradius { \fp_to_dim:N \l__trt_fr_radius_fp }
561 }
562 {6} {
563 \trt_sp_symmedian:nnnn {#1} {#2} {#3}
564 {\tl_use:N \l__trt_fr_output_name_tl}
565 }
566 {7} {
567 \trt_sp_gergonne:nnnn {#1} {#2} {#3}
568 {\tl_use:N \l__trt_fr_output_name_tl}
569 }
570 {8} {
571 \trt_sp_nagel:nnnn {#1} {#2} {#3}

```



```

572         {\tl_use:N \l__trt_fr_output_name_tl}
573     }
574     {9} {
575         \trt_sp_mittenpunkt:nnnn {#1} {#2} {#3}
576         {\tl_use:N \l__trt_fr_output_name_tl}
577     }
578     {10} {
579         \trt_sp_spieker:nnnn {#1} {#2} {#3} {trt@tmp@center}
580         \group_begin:
581             \__trt_bc_initialize:nnn {#1} {#2} {#3}
582             \coordinate (trt@tmp@midi) at (bc3 ~ cs \c_colon_str 1,1,0);
583             \coordinate (trt@tmp@midii) at (bc3 ~ cs \c_colon_str 0,1,1);
584         \group_end:
585         \pgfkeysalso{
586             foot-of~perpendicular=(trt@tmp@center)--(trt@tmp@midi)(trt@tmp@midii)
587         }
588         \trt_distance:nnN {\tl_use:N \l__trt_fr_output_name_tl} {trt@tmp@center}
589         \l__trt_fr_radius_fp
590         \cs_gset_nopar:Npx \trradius { \fp_to_dim:N \l__trt_fr_radius_fp }
591         \coordinate (\tl_use:N \l__trt_fr_output_name_tl) at (trt@tmp@center);
592     }
593 }
594 {
595     \msg_error:nnn {triangletools} {center-not-found} {#4}
596 }
597 }
598 }
599 </frontend>

```

(End definition for *triangle center*. This function is documented on page 4.)

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

B		F	
bc3	8, <u>196</u>	foot_of_perpendicular	6, <u>494</u>
bool commands:		fp commands:	
\bool_if:NTF	198	\fp_compare:nNnTF	101
\bool_new:N	178	\fp_eval:n	291, 292, 293, 303, 304, 305, 403, 404, 405, 415, 416, 417, 428, 429, 430, 441, 447, 453, 469, 470, 471
\bool_set_true:N	182	\fp_new:N	62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 169, 170, 171, 172, 173, 174, 175, 176, 177, 179, 231, 232, 233, 234, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 482
C		\fp_set:Nn	79, 80, 83, 84, 85, 89, 93, 100, 105, 106, 109, 110, 133, 134, 137, 138, 139, 185, 186, 189, 190, 193, 194, 208, 209, 210, 211, 212, 218, 239, 240, 243, 244, 245, 320, 321, 324, 325, 328, 329, 330, 334, 341, 345, 386, 387, 390, 391
centroid	3, <u>505</u>	\fp_to_dim:N	16, 129, 160, 224, 225, 339, 340, 350, 351,
circumcenter	3, <u>505</u>		
\coordinate	128, 159, 338, 349, 366, 392, 511, 519, 556, 582, 583, 591		
cs commands:			
\cs_gset_nopar:Npn	483		
\cs_gset_nopar:Npx	510, 518, 530, 560, 590		
\cs_new:Npn	75, 98, 113, 124, 131, 141, 155, 180, 206, 235, 256, 284, 296, 308, 315, 370, 379, 396, 408, 421, 434, 462		
E			
excenter	3, <u>505</u>		

367, 392, 393, 510, 518, 530, 560, 590	
G	
group commands:	
\group_begin:	
. 286, 298, 310, 317, 372,	
381, 398, 410, 423, 436, 464, 554, 580	
\group_end: 294, 306, 313, 368, 377,	
394, 406, 419, 432, 460, 473, 557, 584	
I	
incenter 3, 505	
initialize_barycentric 8, 502	
\input 9, 10, 11, 12, 13, 14	
int commands:	
\int_case:nnTF 537	
\int_new:N 481	
intersection 6, 494	
M	
msg commands:	
\msg_error:nn 203, 485	
\msg_error:nnn 595	
\msg_new:nnnn 19, 30, 39, 48	
\msg_warning:nnnnnn 103	
N	
\newdimen 7, 8	
O	
orthocenter 4, 505	
output_name 9, 491	
P	
\path 290, 302, 312, 402, 414, 427, 440, 468	
\pgfextractx . . . 77, 81, 135, 183, 187,	
191, 237, 241, 318, 322, 326, 384, 388	
\pgfextracty . . . 78, 82, 136, 184, 188,	
192, 238, 242, 319, 323, 327, 385, 389	
\pgfkeys 489	
\pgfkeysalso 507, 515, 540, 543, 546, 549, 585	
\pgfpointanchor 77, 78, 81, 82,	
135, 136, 183, 184, 187, 188, 191,	
192, 237, 238, 241, 242, 318, 319,	
322, 323, 326, 327, 384, 385, 388, 389	
\protect 56	
\ProvidesExplFile 17, 60, 164, 229, 264, 477	
\ProvidesExplPackage 5	
Q	
quark commands:	
\q_stop 200, 206	
R	
\RequirePackage 3, 4	
S	
\space 56	
str commands:	
\c_colon_str 290, 302, 312,	
402, 414, 427, 440, 468, 556, 582, 583	
T	
TeX and L ^A T _E X 2 _ε commands:	
\pgf@x 224	
\pgf@y 225	
\trt@tmp@ei 7, 77, 79, 81, 83,	
135, 137, 183, 185, 187, 189, 191,	
193, 237, 239, 241, 243, 318, 320,	
322, 324, 326, 328, 384, 386, 388, 390	
\trt@tmp@ei 7, 78, 80, 82, 84,	
136, 138, 184, 186, 188, 190, 192,	
194, 238, 240, 242, 244, 319, 321,	
323, 325, 327, 329, 385, 387, 389, 391	
\tikzdeclarecoordinatesystem 196	
\tikzset 487	
tl commands:	
\tl_new:N 479	
\tl_set:Nn 480, 492	
\tl_use:N 496, 500, 508,	
511, 516, 519, 523, 527, 528, 534,	
553, 558, 564, 568, 572, 576, 588, 591	
triangle_center 4, 536	
trt 2, 487	
trt commands:	
\trt_distance:nnN 9, 9, 235,	
258, 259, 260, 508, 516, 528, 558, 588	
\trt_distance_triangle:nnnNNN 9,	
256, 288, 300, 400, 412, 425, 438, 466	
\trt_lt_get_intersection_-	
coordinate:nnnnNN . . . 7, 113, 126	
\trt_lt_get_intersection_-	
line:NNNNNNNN	
. 7, 7, 12, 98, 119, 150, 362	
\trt_lt_get_line_equation:nnNNN	
. 6, 75, 115, 117, 143, 352, 357	
\trt_lt_get_perpendicular_-	
coordinate:nnnNN . . . 8, 14, 141, 157	
\trt_lt_get_perpendicular_-	
equation:nnnnNNN	
. 7, 131, 146, 354, 359	
\trt_sp_centroid:nnnn . . . 3, 308, 522	
\trt_sp_circumcenter:nnnn	
. 3, 315, 382, 526	
\trt_sp_excenter:nnnn . . . 3, 296, 514	
\trt_sp_gergonne:nnnn . . . 4, 408, 567	
\trt_sp_incenter:nnnn . . . 3, 284, 506	
\trt_sp_mittenpunkt:nnnn . . 5, 434, 575	
\trt_sp_nagel:nnnn 5, 421, 571	
\trt_sp_ninepointcenter:nnnn . . .	
. 4, 379, 552	
\trt_sp_orthocenter:nnnn	
. 4, 370, 383, 533	
\trt_sp_spieker:nnnn 5, 462, 579	
\trt_sp_symmedian:nnnn . . . 4, 396, 563	
trt internal commands:	
\l__trt_bc_anchor_iiix_fp	
. 169, 193, 216	
\l__trt_bc_anchor_iiiy_fp	
. 169, 194, 222	
\l__trt_bc_anchor_iix_fp 169, 189, 215	
\l__trt_bc_anchor_iiy_fp 169, 190, 221	
\l__trt_bc_anchor_ix_fp 169, 185, 214	
\l__trt_bc_anchor_iy_fp 169, 186, 220	
__trt_bc_initialize:nnn	
. 15, 180, 287, 299, 311,	
399, 411, 424, 437, 465, 503, 555, 581	
\l__trt_bc_initialized_bool . . .	
. 15, 178, 182, 198	

\l__trt_bc_lambda_i_fp	\l__trt_lt_tmp_fp
..... 175 , 209 , 214 , 220 12 , 72 , 100 , 101 , 109 , 110
\l__trt_bc_lambda_ii_fp	\l__trt_lt_tmpa_fp
..... 175 , 210 , 215 , 221 72 , 127 , 129 , 137 , 139 , 158 , 160
\l__trt_bc_lambda_iii_fp	\l__trt_lt_tmpb_fp
..... 175 , 211 , 216 , 222 72 , 127 , 129 , 138 , 139 , 158 , 160
__trt_bc_parse:w	\l__trt_sp_a_fp
200 , 206	266 ,
\l__trt_bc_tmp_fp	289 , 291 , 301 , 303 , 401 , 403 , 413 ,
..... 179 , 208 , 209 , 210 , 211	415 , 416 , 417 , 426 , 428 , 429 , 430 ,
\l__trt_d_pointi_x_fp	439 , 443 , 444 , 450 , 456 , 467 , 470 , 471
..... 231 , 239 , 248 , 249	\l__trt_sp_b_fp
\l__trt_d_pointi_y_fp	266 ,
..... 231 , 240 , 251 , 252	289 , 292 , 301 , 304 , 401 , 404 , 413 ,
\l__trt_d_pointii_x_fp	415 , 416 , 417 , 426 , 428 , 429 , 430 ,
..... 231 , 243 , 248 , 249	439 , 444 , 449 , 450 , 456 , 467 , 469 , 471
\l__trt_d_pointii_y_fp	\l__trt_sp_c_fp
..... 231 , 244 , 251 , 252	266 ,
\l__trt_fr_center_number_int ...	289 , 293 , 301 , 305 , 401 , 405 , 413 ,
481	415 , 416 , 417 , 426 , 428 , 429 , 430 ,
\l__trt_fr_output_name_tl	439 , 444 , 450 , 455 , 456 , 467 , 469 , 470
..... 479 , 492 , 496 , 500 , 508 ,	\l__trt_sp_coordinatei_x_fp ...
511 , 516 , 519 , 523 , 527 , 528 , 534 , 269 , 320 , 332 , 343
553 , 558 , 564 , 568 , 572 , 576 , 588 , 591	\l__trt_sp_coordinatei_y_fp ...
\l__trt_fr_radius_fp 482 , 509 , 510 , 269 , 321 , 336 , 347
517 , 518 , 529 , 530 , 559 , 560 , 589 , 590	\l__trt_sp_coordinateii_x_fp ...
\l__trt_i_pointi_x_fp 269 , 324 , 330 , 332 , 339
79 , 91 , 95	\l__trt_sp_coordinateii_y_fp ...
\l__trt_i_pointi_y_fp 269 , 325 , 334 , 336 , 340
80 , 87 , 95	\l__trt_sp_coordinateiii_x_fp ..
\l__trt_i_pointii_x_fp 269 , 328 , 341 , 343 , 350
83 , 91	\l__trt_sp_coordinateiii_y_fp ..
\l__trt_i_pointii_y_fp 269 , 329 , 345 , 347 , 351
84 , 87	\l__trt_sp_linei_a_fp ..
\l__trt_lt_linei_a_fp	269 , 356 , 363
..... 66 , 116 , 120 , 144 , 147 , 151	\l__trt_sp_linei_b_fp ..
\l__trt_lt_linei_b_fp	269 , 356 , 363
..... 66 , 116 , 120 , 144 , 147 , 151	\l__trt_sp_linei_c_fp ..
\l__trt_lt_linei_c_fp	269 , 356 , 363
..... 66 , 116 , 120 , 144 , 147 , 151	\l__trt_sp_lineii_a_fp ..
\l__trt_lt_lineii_a_fp	269 , 361 , 364
..... 66 , 118 , 121 , 148 , 152	\l__trt_sp_lineii_b_fp ..
\l__trt_lt_lineii_b_fp	269 , 361 , 364
..... 66 , 118 , 121 , 148 , 152	\l__trt_sp_lineii_c_fp ..
\l__trt_lt_lineii_c_fp	269 , 361 , 364
..... 66 , 118 , 121 , 148 , 152	\l__trt_sp_tmpa_fp
\l__trt_lt_pointi_x_fp	281 , 353 ,
62	355 , 358 , 360 , 365 , 367 , 386 , 390 , 392
\l__trt_lt_pointi_y_fp	\l__trt_sp_tmpb_fp
62	281 , 353 ,
\l__trt_lt_pointii_x_fp	355 , 358 , 360 , 365 , 367 , 387 , 391 , 393
62	\l__trt_sp_tmpc_fp
\l__trt_lt_pointii_y_fp 281 , 353 , 355 , 358 , 360
62	\l__trt_tmp_a_fp
__trt_lt_return_intersection:nnnnn	212 , 224
..... 124 , 375 , 495	\l__trt_tmp_b_fp
__trt_lt_return_perpendicular_-	218 , 225
coordinate:nnnn 155 , 373 , 374 , 499	\trradius
	2 ,
	11 , 56 , 482 , 510 , 518 , 530 , 560 , 590