

The triangletools package

Vu Van Dung

May 7, 2020

Contents

1	Introduction	1
2	Loading the package	2
3	User interface	2
3.1	Accessing the keys	2
3.2	Triangle centers	2
3.3	Other utilities	5
3.3.1	Line tools	5
3.3.2	The barycentric coordinate system	7
3.3.3	Distance-finding utility	8
3.4	Customization	8
4	Implementation	8
4.1	The main package file	8
4.2	Errors and warnings	9
4.3	The backend layer	10
4.3.1	The line tools utility	10
4.3.2	The barycentric coordinate system utility	12
4.3.3	Distance-finding utility	14
4.4	Construction of triangle centers	15
4.4.1	X_1 – The incenter	16
4.4.2	X_2 – The centroid	17
4.4.3	X_3 – The circumcenter	17
4.4.4	X_4 – The orthocenter	18
4.4.5	X_5 – The nine-point center	18
4.4.6	X_6 – The symmedian point	19
4.4.7	X_7 – The Gergonne point	19
4.4.8	X_8 – The Nagel point	20
4.4.9	X_9 – The <i>mittenpunkt</i>	20
4.4.10	X_{10} – The Spieker point	20
4.5	The frontend layer	21
	Index	23

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 3.3) 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

/tikz/trt /tikz/trt={⟨keys⟩}

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

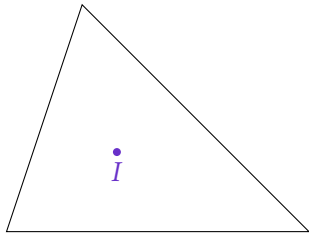
This key is used to access all other keys in the user interface.

3.2 Triangle centers

incenter /tikz/triangletools/incenter=(⟨coord 1⟩)(⟨coord 2⟩)(⟨coord 3⟩)
\trt_sp_incenter:nnnn \trt_sp_incenter:nnnn {⟨coord 1⟩}{⟨coord 2⟩}{⟨coord 3⟩}{⟨name⟩}

Find the incenter X_1 of the triangle joining TikZ coordinates ⟨coord 1⟩, ⟨coord 2⟩ and ⟨coord 3⟩. The incenter is saved to TikZ coordinate ⟨name⟩.

If you use the key (why do you use the function anyway), ⟨name⟩ is set to trt output by default. You can change that using output name, see Section 3.4.



```

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 \end{tikzpicture}

```

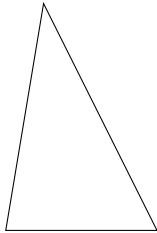
excenter

```

/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: $\text{excenter}=(a)(b)(c)$ is *different* from $\text{excenter}=(b)(a)(c)$.



```

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 \end{tikzpicture}

```

I_a

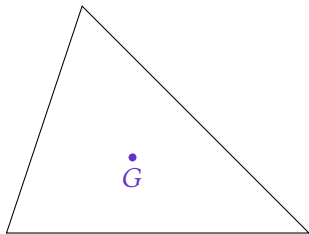
centroid

```

/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}

```

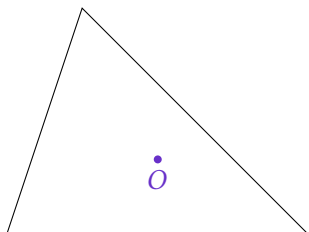
circumcenter

```

/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 \end{tikzpicture}

```

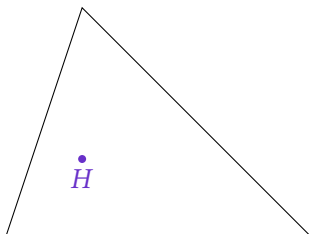
orthocenter

```

/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`

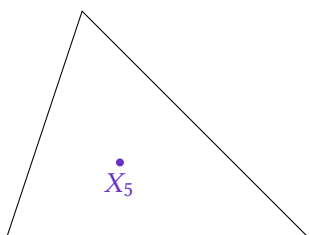
`/tikz/triangletools/triangle center=($\langle coor 1 \rangle$)($\langle coor 2 \rangle$)($\langle coor 3 \rangle$)($\langle index \rangle$)`

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

`\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.



```

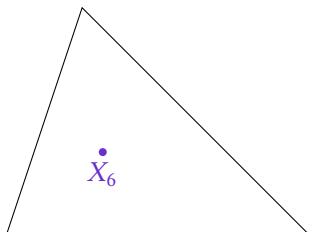
1 \begin {tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={triangle center=(a)(b)(c)(5)}] (trt output)
6     circle[radius=1.5pt] node[below] {$X_5$};
7 \end{tikzpicture}

```

`\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.



```

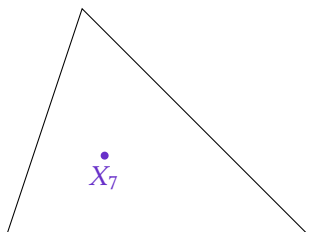
1 \begin {tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={triangle center=(a)(b)(c)(6)}] (trt output)
6     circle[radius=1.5pt] node[below] {$X_6$};
7 \end{tikzpicture}

```

`\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.



```

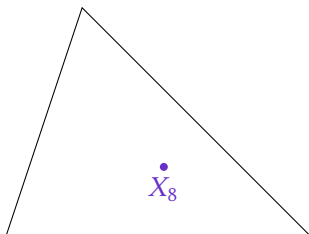
1 \begin {tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={triangle center=(a)(b)(c)(7)}] (trt output)
6     circle[radius=1.5pt] node[below] {$X_7$};
7 \end{tikzpicture}

```

`\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}

```

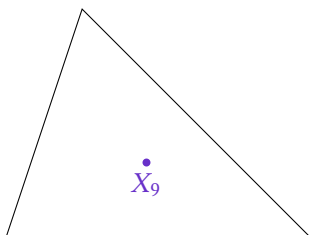
`\trt_sp_mittenpunkt:nnnn`

```

/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}

```

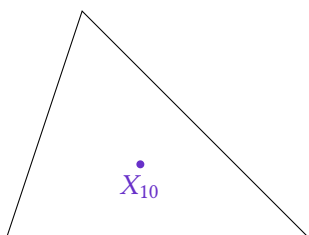
`\trt_sp_spieker:nnnn`

```

/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 \end{tikzpicture}

```

3.3 Other utilities

3.3.1 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.



```

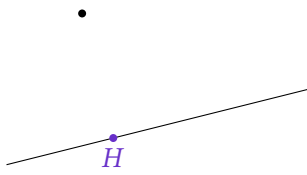
1 \begin{tikzpicture}
2   \draw (0,0) coordinate (a) -- (0,1) coordinate (b);
3   \draw (1,0) coordinate (c) -- (.5,1) coordinate (d);
4   \coordinate (e) at (1,1);
5   \fill[maincolor,trt={intersection=(a)(b)--(c)(d)}] (trt output)
6     circle (1.5pt) node[above] {$I$};
7   \fill[maincolor,trt={intersection=(a)(b)--(c)(e)}]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

```

foot_of_perpendicular

/tikz/triangletools/foot of perpendicular= $(\langle \text{coord } 1 \rangle) -- (\langle \text{coord } 2 \rangle) (\langle \text{coord } 3 \rangle)$

Find the foot of perpendicular of point $\langle \text{coord } 1 \rangle$ to the line joining $\langle \text{coord } 2 \rangle$ and $\langle \text{coord } 3 \rangle$.



```

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 $\langle \text{coord } 1 \rangle \langle \text{coord } 2 \rangle \langle a \rangle \langle b \rangle \langle c \rangle$

Find the equation of the line joining $\langle \text{coord } 1 \rangle$ and $\langle \text{coord } 2 \rangle$, 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 $\langle \text{coord } 1 \rangle$ and $\langle \text{coord } 2 \rangle$, 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.

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

```

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  \fp_new:N \c_foo_cm_fp
11  \fp_set:Nn \c_foo_cm_fp {28.45275590551181}
12  \def \resultequation {
13    $\fp_eval:n {round (\l_foo_tmpa_fp / \c_foo_cm_fp)}x +
14    \fp_eval:n {round (\l_foo_tmpb_fp / \c_foo_cm_fp)}y
15    =\fp_eval:n {round (\l_foo_tmpc_fp / (\c_foo_cm_fp * \c_foo_cm_fp))}$
16  }
17  \ExplSyntaxOff
18  \draw (a) -- (b) node[midway,sloped,below] {\resultequation};
19 \end{tikzpicture}

```

\trt_lt_get_intersection_line:NNNNNNNN

\trt_lt_get_intersection_line:NNNNNNNN $\langle a1 \rangle \langle b1 \rangle \langle c1 \rangle \langle a2 \rangle \langle b2 \rangle \langle c2 \rangle \langle x \rangle \langle y \rangle$

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 $\langle \text{coord } 1 \rangle \langle \text{coord } 2 \rangle \langle \text{coord } 3 \rangle \langle \text{coord } 4 \rangle \langle x \rangle \langle y \rangle$

This function is a wrapper of `\trt_lt_get_intersection_line:NNNNNNNN`. It finds the intersection of the line joining $\langle \text{coord } 1 \rangle$, $\langle \text{coord } 2 \rangle$ and the line joining $\langle \text{coord } 3 \rangle$, $\langle \text{coord } 4 \rangle$. 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:nnnnN {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}

```

 $\backslash\mathrm{trt_lt_get_perpendicular_equation:nnNNNNN}$

 $\backslash\mathrm{trt_lt_get_perpendicular_equation:nNNNNNN}\{\langle\mathrm{coord}\ 1\rangle\}\langle a1\rangle\langle b1\rangle\langle c1\rangle\langle a2\rangle\langle b2\rangle\langle c2\rangle$

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

```

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:nnNNNNN {a}
15 \l_foo_tmpe_fp \l_foo_tmpe_fp \l_foo_tmpe_fp
16 \l_foo_tmpe_fp \l_foo_tmpe_fp \l_foo_tmpe_fp
17 \fp_new:N \c_foo_cm_fp
18 \fp_set:Nn \c_foo_cm_fp {28.45275590551181}
19 \def \resultequation {
20 $\fp_eval:n {\round (\l_foo_tmpe_fp / \c_foo_cm_fp)}x +
21 \fp_eval:n {\round (\l_foo_tmpe_fp / \c_foo_cm_fp)}y
22 =\fp_eval:n {\round (\l_foo_tmpe_fp / (\c_foo_cm_fp * \c_foo_cm_fp))}$
23 }
24 \ExplSyntaxOff
25 \path (1,0) node[below=3mm,align=center]
26 {Equation of perpendicular line:\\\resultequation};
27 \end{tikzpicture}

```

(0.5, 4)

•

(3, 1)

(-1, 0)

Equation of perpendicular line:
 $4x + 1y = 6$

 $\backslash\mathrm{trt_lt_get_perpendicular_coordinate:nnnNN}$

 $\backslash\mathrm{trt_lt_get_perpendicular_coordinate:nnnNN}\{\langle\mathrm{coord}\ 1\rangle\}\{\langle\mathrm{coord}\ 2\rangle\}\{\langle\mathrm{coord}\ 3\rangle\}\langle x\rangle\langle y\rangle$

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

This is the base of foot of perpendicular.

3.3.2 The barycentric coordinate system

 $\mathrm{initialize_barycentric}$

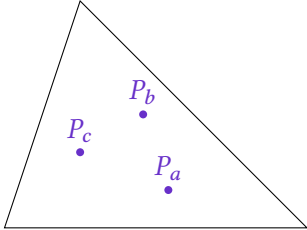
 $\mathrm{/tikz/triangletools/initialize\ barycentric}=(\langle\mathrm{coord}\ 1\rangle)(\langle\mathrm{coord}\ 2\rangle)(\langle\mathrm{coord}\ 3\rangle)$

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

bc3 (bc3 cs:⟨l1⟩,⟨l2⟩,⟨l3⟩)

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 ⟨l1⟩, ⟨l2⟩ and ⟨l3⟩ is not necessarily 1 – the package will take care of that internally.



```

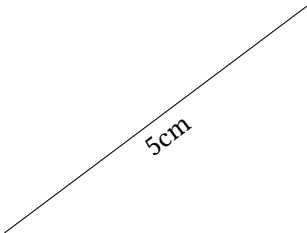
1 \begin{tikzpicture}
2   \draw (1,3) coordinate (a) --
3         (0,0) coordinate (b) --
4         (4,0) coordinate (c) -- cycle;
5   \fill[maincolor,trt={initialize barycentric=(a)(b)(c)}] (bc3 cs:1,2,3)
6     circle[radius=1.5pt] node[above] {$P_a$};
7   \fill[maincolor,trt={initialize barycentric=(b)(c)(a)}]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

```

3.3.3 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   \fp_new:N \c_foo_cm_fp
7   \fp_set:Nn \c_foo_cm_fp {28.45275590551181}
8   \draw (a) -- (b) node[midway,sloped,below]
9     { \fp_eval:n {round(\l_foo_tmpa_fp / \c_foo_cm_fp)} cm };
10  \ExplSyntaxOff
11 \end{tikzpicture}

```

`\trt_distance_triangle:nnnNNN` `\trt_distance_triangle:nnnNNN{⟨coord 1⟩}{⟨coord 2⟩}{⟨coord 3⟩}{⟨a⟩}{⟨b⟩}{⟨c⟩}`

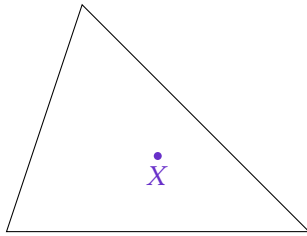
`\trt_distance:nnN` is needed to find the side lengths in a triangle (these side lengths are very helpful in many areas, for instance in this package to find special points based on the barycentric system). However, using it three times in a row is not quite elegant; this function is defined to automate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

⟨a⟩ is set to the distance between ⟨coord 2⟩ and ⟨coord 3⟩, similar things happen for ⟨b⟩ and ⟨c⟩.

3.4 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 }
48 </messages>

```

4.3 The backend layer

4.3.1 The line tools utility

```

49 <*linetools>
50 \ProvidesExplFile {trtlinetools.code.tex} {2020/04/30} {0.1}
51 {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

```

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

```

52 \fp_new:N \__trt_lt_pointi_x_fp
53 \fp_new:N \__trt_lt_pointi_y_fp
54 \fp_new:N \__trt_lt_pointii_x_fp
55 \fp_new:N \__trt_lt_pointii_y_fp

```

(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

```

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.

```

56 \fp_new:N \__trt_lt_linei_a_fp
57 \fp_new:N \__trt_lt_linei_b_fp
58 \fp_new:N \__trt_lt_linei_c_fp
59 \fp_new:N \__trt_lt_lineii_a_fp
60 \fp_new:N \__trt_lt_lineii_b_fp
61 \fp_new:N \__trt_lt_lineii_c_fp

```

(End definition for `__trt_lt_linei_a_fp` and others.)

```

\__trt_lt_tmp_fp
\__trt_lt_tmpa_fp
\__trt_lt_tmppb_fp

```

Some additional temporary variables.

```

62 \fp_new:N \__trt_lt_tmp_fp

```

```

63 \fp_new:N \l__trt_lt_tmpa_fp
64 \fp_new:N \l__trt_lt_tmpb_fp

```

(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.

```

65 \cs_new:Npn \trt_lt_get_line_equation:nnNNN #1 #2 #3 #4 #5
66 {
67   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#1}{center}}
68   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#1}{center}}
69   \fp_set:Nn \l__trt_i_pointi_x_fp {\trt@tmp@i}
70   \fp_set:Nn \l__trt_i_pointi_y_fp {\trt@tmp@ii}
71   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#2}{center}}
72   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#2}{center}}
73   \fp_set:Nn \l__trt_i_pointii_x_fp {\trt@tmp@i}
74   \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.

```

75   \fp_set:Nn #3
76   {
77     \l__trt_i_pointii_y_fp - \l__trt_i_pointi_y_fp
78   }
79   \fp_set:Nn #4
80   {
81     \l__trt_i_pointi_x_fp - \l__trt_i_pointii_x_fp
82   }
83   \fp_set:Nn #5
84   {
85     #3 * \l__trt_i_pointi_x_fp + #4 * \l__trt_i_pointi_y_fp
86   }
87 }

```

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

`\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.

```

88 \cs_new:Npn \trt_lt_get_intersection_line:NNNNNNNN #1 #2 #3 #4 #5 #6 #7 #8
89 {
90   \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.

```

91   \fp_compare:nNnTF {\l__trt_lt_tmp_fp} = {0}
92   {
93     \msg_warning:nnnnnn {triangletools} {intersection-not-found}
94     {(#1)} {(#2)} {(#3)} {(#4)}
95     \fp_set:Nn #7 {0}
96     \fp_set:Nn #8 {0}
97   }
98   {
99     \fp_set:Nn #7 { ( #5 * #3 - #2 * #6 ) / \l__trt_lt_tmp_fp }
100    \fp_set:Nn #8 { ( #1 * #6 - #4 * #3 ) / \l__trt_lt_tmp_fp }
101  }
102 }

```

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

`\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.

```

103 \cs_new:Npn \trt_lt_get_intersection_coordinate:nnnnNN #1 #2 #3 #4 #5 #6
104 {
105   \trt_lt_get_line_equation:nnNNN {#1} {#2}
106   \l__trt_lt_line_i_a_fp \l__trt_lt_line_i_b_fp \l__trt_lt_line_i_c_fp
107   \trt_lt_get_line_equation:nnNNN {#3} {#4}
108   \l__trt_lt_line_ii_a_fp \l__trt_lt_line_ii_b_fp \l__trt_lt_line_ii_c_fp
109   \trt_lt_get_intersection_line:NNNNNNNN
110   \l__trt_lt_line_i_a_fp \l__trt_lt_line_i_b_fp \l__trt_lt_line_i_c_fp
111   \l__trt_lt_line_ii_a_fp \l__trt_lt_line_ii_b_fp \l__trt_lt_line_ii_c_fp
112   #5 #6
113 }

```

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

`__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.

```

114 \cs_new:Npn \__trt_lt_return_intersection:nnnnn #1 #2 #3 #4 #5
115 {
116   \trt_lt_get_intersection_coordinate:nnnnNN {#1} {#2} {#3} {#4}
117   \l__trt_lt_tmpa_fp \l__trt_lt_tmpb_fp
118   \coordinate[overlay] (#5) at
119     (\fp_to_dim:N \l__trt_lt_tmpa_fp, \fp_to_dim:N \l__trt_lt_tmpb_fp);
120 }

```

(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.

```

121 \cs_new:Npn \trt_lt_get_perpendicular_equation:nNNNNNN #1 #2 #3 #4 #5 #6 #7
122 {
123   \fp_set:Nn #5 { -#3 }
124   \fp_set:Nn #6 { #2 }
125   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#1}{center}}
126   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#1}{center}}
127   \fp_set:Nn \l__trt_lt_tmpa_fp {\trt@tmp@i}
128   \fp_set:Nn \l__trt_lt_tmpb_fp {\trt@tmp@ii}
129   \fp_set:Nn #7 { #5 * \l__trt_lt_tmpa_fp + #6 * \l__trt_lt_tmpb_fp }
130 }

```

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

`\trt_lt_get_perpendicular_coordinate:nnnNN` The base implemented, let's find the foot of perpendicular from a point to a segment.

```

131 \cs_new:Npn \trt_lt_get_perpendicular_coordinate:nnnNN #1 #2 #3 #4 #5
132 {
133   \trt_lt_get_line_equation:nnNNN {#2} {#3}
134   \l__trt_lt_line_i_a_fp \l__trt_lt_line_i_b_fp \l__trt_lt_line_i_c_fp
135
136   \trt_lt_get_perpendicular_equation:nNNNNNN {#1}
137   \l__trt_lt_line_i_a_fp \l__trt_lt_line_i_b_fp \l__trt_lt_line_i_c_fp
138   \l__trt_lt_line_ii_a_fp \l__trt_lt_line_ii_b_fp \l__trt_lt_line_ii_c_fp

```

```

139
140 \trt_lt_get_intersection_line:nnnnNNN
141 \l__trt_lt_linei_a_fp \l__trt_lt_linei_b_fp \l__trt_lt_linei_c_fp
142 \l__trt_lt_lineii_a_fp \l__trt_lt_lineii_b_fp \l__trt_lt_lineii_c_fp
143 #4 #5
144 }

```

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

`\trt_lt_return_perpendicular_coordinate:nnnn`

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

```

145 \cs_new:Npn \__trt_lt_return_perpendicular_coordinate:nnnn #1 #2 #3 #4
146 {
147   \trt_lt_get_perpendicular_coordinate:nnnNN {#1} {#2} {#3}
148   \l__trt_lt_tmpa_fp \l__trt_lt_tmpb_fp
149   \coordinate[overlay] (#4) at
150   (\fp_to_dim:N \l__trt_lt_tmpa_fp, \fp_to_dim:N \l__trt_lt_tmpb_fp);
151 }
152 </linetools>

```

(End definition for `__trt_lt_return_perpendicular_coordinate:nnnn`.)

4.3.2 The barycentric coordinate system utility

```

153 <*barycentric>
154 \ProvidesExplFile {trtbarycentric.code.tex} {2020/04/30} {0.1}
155 {
156   The ~ triangletools ~ package: ~ Utilities ~ for ~ the ~ barycentric ~
157   coordinate ~ system.
158 }

```

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.

```

159 \fp_new:N \l__trt_bc_anchor_ix_fp
160 \fp_new:N \l__trt_bc_anchor_iy_fp
161 \fp_new:N \l__trt_bc_anchor_iix_fp
162 \fp_new:N \l__trt_bc_anchor_iiy_fp
163 \fp_new:N \l__trt_bc_anchor_iiix_fp
164 \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.

```

165 \fp_new:N \l__trt_bc_lambda_i_fp
166 \fp_new:N \l__trt_bc_lambda_ii_fp
167 \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.

```

168 \bool_new:N \l__trt_bc_initialized_bool

```

(End definition for `\l__trt_bc_initialized_bool`.)

`__trt_bc_tmp_fp` A temporary variable.

```
169 \fp_new:N \__trt_bc_tmp_fp
```

(End definition for `__trt_bc_tmp_fp`.)

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

```
170 \cs_new:Npn \__trt_bc_initialize:nnn #1 #2 #3
171 {
172   \bool_set_true:N \__trt_bc_initialized_bool
173   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#1}{center}}
174   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#1}{center}}
175   \fp_set:Nn \__trt_bc_anchor_ix_fp {\trt@tmp@i}
176   \fp_set:Nn \__trt_bc_anchor_iy_fp {\trt@tmp@ii}
177   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#2}{center}}
178   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#2}{center}}
179   \fp_set:Nn \__trt_bc_anchor_iix_fp {\trt@tmp@i}
180   \fp_set:Nn \__trt_bc_anchor_iiy_fp {\trt@tmp@ii}
181   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#3}{center}}
182   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#3}{center}}
183   \fp_set:Nn \__trt_bc_anchor_iiix_fp {\trt@tmp@i}
184   \fp_set:Nn \__trt_bc_anchor_iiiy_fp {\trt@tmp@ii}
185 }
```

(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.

```
186 \tikzdeclarecoordinatesystem {bc3}
187 {
188   \bool_if:NTF \__trt_bc_initialized_bool
189   {
190     \__trt_bc_parse:w #1 \q_stop
191   }
192   {
193     \msg_error:nn {triangletools} {uninitialized}
194   }
195 }
```

(End definition for bc3. This function is documented on page 7.)

`__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 $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

```
196 \cs_new:Npn \__trt_bc_parse:w #1,#2,#3 \q_stop
197 {
198   \fp_set:Nn \__trt_bc_tmp_fp { (#1) + (#2) + (#3) }
199   \fp_set:Nn \__trt_bc_lambda_i_fp { (#1) / (\__trt_bc_tmp_fp) }
200   \fp_set:Nn \__trt_bc_lambda_ii_fp { (#2) / (\__trt_bc_tmp_fp) }
201   \fp_set:Nn \__trt_bc_lambda_iii_fp { (#3) / (\__trt_bc_tmp_fp) }
202   \fp_set:Nn \__trt_tmp_a_fp
203   {
204     \__trt_bc_anchor_ix_fp * \__trt_bc_lambda_i_fp +
```

```

205         \l__trt_bc_anchor_iix_fp * \l__trt_bc_lambda_ii_fp +
206         \l__trt_bc_anchor_iiix_fp * \l__trt_bc_lambda_iii_fp
207     }
208     \fp_set:Nn \l__trt_tmp_b_fp
209     {
210         \l__trt_bc_anchor_iy_fp * \l__trt_bc_lambda_i_fp +
211         \l__trt_bc_anchor_iiy_fp * \l__trt_bc_lambda_ii_fp +
212         \l__trt_bc_anchor_iiiy_fp * \l__trt_bc_lambda_iii_fp
213     }

```

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

```

214     \pgf@x = \fp_to_dim:N \l__trt_tmp_a_fp
215     \pgf@y = \fp_to_dim:N \l__trt_tmp_b_fp
216 }
217 </barycentric>

```

(End definition for `__trt_bc_parse:w`.)

4.3.3 Distance-finding utility

```

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

```

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

```

\l__trt_d_pointi_x_fp
\l__trt_d_pointi_y_fp
\l__trt_d_pointii_x_fp
\l__trt_d_pointii_y_fp

```

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

```

221 \fp_new:N \l__trt_d_pointi_x_fp
222 \fp_new:N \l__trt_d_pointii_x_fp
223 \fp_new:N \l__trt_d_pointi_y_fp
224 \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.

```

225 \cs_new:Npn \trt_distance:nnN #1 #2 #3
226 {
227     \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#1}{center}}
228     \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#1}{center}}
229     \fp_set:Nn \l__trt_d_pointi_x_fp {\trt@tmp@i}
230     \fp_set:Nn \l__trt_d_pointi_y_fp {\trt@tmp@ii}
231     \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#2}{center}}
232     \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#2}{center}}
233     \fp_set:Nn \l__trt_d_pointii_x_fp {\trt@tmp@i}
234     \fp_set:Nn \l__trt_d_pointii_y_fp {\trt@tmp@ii}
235     \fp_set:Nn #3
236     {
237         sqrt((
238             (\l__trt_d_pointi_x_fp - \l__trt_d_pointii_x_fp) *
239             (\l__trt_d_pointi_x_fp - \l__trt_d_pointii_x_fp)
240         ) + (
241             (\l__trt_d_pointi_y_fp - \l__trt_d_pointii_y_fp) *
242             (\l__trt_d_pointi_y_fp - \l__trt_d_pointii_y_fp)
243         ))
244     }
245 }

```

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

`\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.

```

246 \cs_new:Npn \trt_distance_triangle:nnnNNN #1 #2 #3 #4 #5 #6
247 {
248   \trt_distance:nnN {#2} {#3} #4
249   \trt_distance:nnN {#3} {#1} #5
250   \trt_distance:nnN {#1} {#2} #6
251 }
252 </distance>

```

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

4.4 Construction of triangle centers

```

253 <*specialpoints>
254 \ProvidesExplFile {trtspecialpoints.code.tex} {2020/04/30} {0.1}
255 {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.

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

```

\l__trt_sp_a_fp
\l__trt_sp_b_fp
\l__trt_sp_c_fp
256 \fp_new:N \l__trt_sp_a_fp
257 \fp_new:N \l__trt_sp_b_fp
258 \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`.)

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_x_fp
\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
\l__trt_sp_linei_a_fp
\l__trt_sp_linei_b_fp
\l__trt_sp_linei_c_fp
\l__trt_sp_lineii_a_fp
\l__trt_sp_lineii_b_fp
\l__trt_sp_lineii_c_fp
259 \fp_new:N \l__trt_sp_coordinatei_x_fp
260 \fp_new:N \l__trt_sp_coordinatei_y_fp
261 \fp_new:N \l__trt_sp_coordinateii_x_fp
262 \fp_new:N \l__trt_sp_coordinateii_y_fp
263 \fp_new:N \l__trt_sp_coordinateiii_x_fp
264 \fp_new:N \l__trt_sp_coordinateiii_y_fp
265 \fp_new:N \l__trt_sp_linei_a_fp
266 \fp_new:N \l__trt_sp_linei_b_fp
267 \fp_new:N \l__trt_sp_linei_c_fp
268 \fp_new:N \l__trt_sp_lineii_a_fp
269 \fp_new:N \l__trt_sp_lineii_b_fp
270 \fp_new:N \l__trt_sp_lineii_c_fp

```

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

Some additional temporary variables.

```

\l__trt_sp_tmpa_fp
\l__trt_sp_tmppb_fp
\l__trt_sp_tmppc_fp
271 \fp_new:N \l__trt_sp_tmpa_fp
272 \fp_new:N \l__trt_sp_tmppb_fp
273 \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) .

```

274 \cs_new:Npn \trt_sp_incenter:nnnn #1 #2 #3 #4
275 {
276   \group_begin:
277     \__trt_bc_initialize:nnn {#1} {#2} {#3}
278     \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
279     \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
280     \path[overlay] (bc3 ~ cs \c_colon_str
281       \fp_eval:n {\l__trt_sp_a_fp},
282       \fp_eval:n {\l__trt_sp_b_fp},
283       \fp_eval:n {\l__trt_sp_c_fp}) coordinate (#4);
284   \group_end:
285 }

```

(End definition for \trt_sp_incenter:nnnn. This function is documented on page 2.)

`\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.

```

286 \cs_new:Npn \trt_sp_excenter:nnnn #1 #2 #3 #4
287 {
288   \group_begin:
289     \__trt_bc_initialize:nnn {#1} {#2} {#3}
290     \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
291     \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
292     \path[overlay] (bc3 ~ cs \c_colon_str
293       \fp_eval:n {- \l__trt_sp_a_fp},
294       \fp_eval:n {\l__trt_sp_b_fp},
295       \fp_eval:n {\l__trt_sp_c_fp}) coordinate (#4);
296   \group_end:
297 }

```

(End definition for \trt_sp_excenter:nnnn. This function is documented on page 2.)

4.4.2 X_2 – The centroid

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

```

298 \cs_new:Npn \trt_sp_centroid:nnnn #1 #2 #3 #4
299 {
300   \group_begin:
301     \__trt_bc_initialize:nnn {#1} {#2} {#3}
302     \path[overlay] (bc3 ~ cs \c_colon_str 1, 1, 1) coordinate (#4);
303   \group_end:
304 }

```

(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.

```

305 \cs_new:Npn \trt_sp_circumcenter:nnnn #1 #2 #3 #4
306 {
307   \group_begin:

```

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

```

308 \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#1}{center}}
309 \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#1}{center}}
310 \fp_set:Nn \l__trt_sp_coordinatei_x_fp {\trt@tmp@i}
311 \fp_set:Nn \l__trt_sp_coordinatei_y_fp {\trt@tmp@ii}
312 \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#2}{center}}
313 \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#2}{center}}
314 \fp_set:Nn \l__trt_sp_coordinateii_x_fp {\trt@tmp@i}
315 \fp_set:Nn \l__trt_sp_coordinateii_y_fp {\trt@tmp@ii}
316 \pgfextractx {\trt@tmp@i} {\pgfpointanchor{#3}{center}}
317 \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{#3}{center}}
318 \fp_set:Nn \l__trt_sp_coordinateiii_x_fp {\trt@tmp@i}
319 \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.

```

320 \fp_set:Nn \l__trt_sp_coordinateii_x_fp
321 {
322   (\l__trt_sp_coordinatei_x_fp + \l__trt_sp_coordinateii_x_fp) / 2
323 }
324 \fp_set:Nn \l__trt_sp_coordinateii_y_fp
325 {
326   (\l__trt_sp_coordinatei_y_fp + \l__trt_sp_coordinateii_y_fp) / 2
327 }
328 \coordinate[overlay] (trt@tmp@ii) at (
329   \fp_to_dim:N \l__trt_sp_coordinateii_x_fp,
330   \fp_to_dim:N \l__trt_sp_coordinateii_y_fp);
331 \fp_set:Nn \l__trt_sp_coordinateiii_x_fp
332 {
333   (\l__trt_sp_coordinatei_x_fp + \l__trt_sp_coordinateiii_x_fp) / 2
334 }
335 \fp_set:Nn \l__trt_sp_coordinateiii_y_fp
336 {
337   (\l__trt_sp_coordinatei_y_fp + \l__trt_sp_coordinateiii_y_fp) / 2
338 }
339 \coordinate[overlay] (trt@tmp@iii) at (
340   \fp_to_dim:N \l__trt_sp_coordinateiii_x_fp,
341   \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.

```

342 \trt_lt_get_line_equation:nnNNN {#1} {#2}
343   \l__trt_sp_tmpa_fp \l__trt_sp_tmpb_fp \l__trt_sp_tmpc_fp
344 \trt_lt_get_perpendicular_equation:nnNNNNN {trt@tmp@ii}
345   \l__trt_sp_tmpa_fp \l__trt_sp_tmpb_fp \l__trt_sp_tmpc_fp
346   \l__trt_sp_linei_a_fp \l__trt_sp_linei_b_fp \l__trt_sp_linei_c_fp
347 \trt_lt_get_line_equation:nnNNN {#1} {#3}
348   \l__trt_sp_tmpa_fp \l__trt_sp_tmpb_fp \l__trt_sp_tmpc_fp
349 \trt_lt_get_perpendicular_equation:nnNNNNN {trt@tmp@iii}
350   \l__trt_sp_tmpa_fp \l__trt_sp_tmpb_fp \l__trt_sp_tmpc_fp
351   \l__trt_sp_lineii_a_fp \l__trt_sp_lineii_b_fp \l__trt_sp_lineii_c_fp
352 \trt_lt_get_intersection_line:nnNNNNNN
353   \l__trt_sp_linei_a_fp \l__trt_sp_linei_b_fp \l__trt_sp_linei_c_fp
354   \l__trt_sp_lineii_a_fp \l__trt_sp_lineii_b_fp \l__trt_sp_lineii_c_fp
355   \l__trt_sp_tmpa_fp \l__trt_sp_tmpb_fp
356 \coordinate[overlay] (#4) at (
357   \fp_to_dim:N \l__trt_sp_tmpa_fp, \fp_to_dim:N \l__trt_sp_tmpb_fp);
358 \group_end:
359 }

```

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.

```

360 \cs_new:Npn \trt_sp_orthocenter:nnnn #1 #2 #3 #4
361 {
362   \group_begin:
363     \__trt_lt_return_perpendicular_coordinate:nnnn {#1} {#2} {#3} {trt@tmp@i}
364     \__trt_lt_return_perpendicular_coordinate:nnnn {#2} {#1} {#3} {trt@tmp@ii}
365     \__trt_lt_return_intersection:nnnnn
366       {#1} {trt@tmp@i} {#2} {trt@tmp@ii} {#4}
367   \group_end:
368 }
```

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

4.4.5 X_5 – The nine-point center

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

```

369 \cs_new:Npn \trt_sp_ninepointcenter:nnnn #1 #2 #3 #4
370 {
371   \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.

```

372   \trt_sp_circumcenter:nnnn {#1} {#2} {#3} {trt@tmp@iii}
373   \trt_sp_orthocenter:nnnn {#1} {#2} {#3} {trt@tmp@iv}
374   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{trt@tmp@iii}{center}}
375   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{trt@tmp@iii}{center}}
376   \fp_set:Nn \__trt_sp_tmpa_fp {\trt@tmp@i}
377   \fp_set:Nn \__trt_sp_tmpb_fp {\trt@tmp@ii}
378   \pgfextractx {\trt@tmp@i} {\pgfpointanchor{trt@tmp@iv}{center}}
379   \pgfextracty {\trt@tmp@ii} {\pgfpointanchor{trt@tmp@iv}{center}}
380   \fp_set:Nn \__trt_sp_tmpa_fp { (\trt@tmp@i + \__trt_sp_tmpa_fp) / 2 }
381   \fp_set:Nn \__trt_sp_tmpb_fp { (\trt@tmp@ii + \__trt_sp_tmpb_fp) / 2 }
382   \coordinate[overlay] (#4) at (\fp_to_dim:N \__trt_sp_tmpa_fp,
383     \fp_to_dim:N \__trt_sp_tmpb_fp);
384   \group_end:
385 }
```

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

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) .

```

386 \cs_new:Npn \trt_sp_symmedian:nnnn #1 #2 #3 #4
387 {
388   \group_begin:
389     \__trt_bc_initialize:nnn {#1} {#2} {#3}
390     \trt_distance_triangle:nnnn {#1} {#2} {#3}
391     \__trt_sp_a_fp \__trt_sp_b_fp \__trt_sp_c_fp
392     \path[overlay] (bc3 ~ cs \c_colon_str
393       \fp_eval:n {\__trt_sp_a_fp * \__trt_sp_a_fp},
394       \fp_eval:n {\__trt_sp_b_fp * \__trt_sp_b_fp},
```

```

395         \fp_eval:n {\l__trt_sp_c_fp * \l__trt_sp_c_fp} coordinate (#4);
396     \group_end:
397 }

```

(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 $(\frac{1}{b+c-a}, \frac{1}{c+a-b}, \frac{1}{a+b-c})$.

```

398 \cs_new:Npn \trt_sp_gergonne:nnnn #1 #2 #3 #4
399 {
400     \group_begin:
401         \__trt_bc_initialize:nnn {#1} {#2} {#3}
402         \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
403         \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
404         \path[overlay] (bc3 ~ cs \c_colon_str
405             \fp_eval:n { 1/(\l__trt_sp_b_fp + \l__trt_sp_c_fp - \l__trt_sp_a_fp) },
406             \fp_eval:n { 1/(\l__trt_sp_c_fp + \l__trt_sp_a_fp - \l__trt_sp_b_fp) },
407             \fp_eval:n { 1/(\l__trt_sp_a_fp + \l__trt_sp_b_fp - \l__trt_sp_c_fp) }
408             ) coordinate (#4);
409     \group_end:
410 }

```

(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 $(b+c-a, c+a-b, a+b-c)$.

```

411 \cs_new:Npn \trt_sp_nagel:nnnn #1 #2 #3 #4
412 {
413     \group_begin:
414         \__trt_bc_initialize:nnn {#1} {#2} {#3}
415         \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
416         \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
417         \path[overlay] (bc3 ~ cs \c_colon_str
418             \fp_eval:n { \l__trt_sp_b_fp + \l__trt_sp_c_fp - \l__trt_sp_a_fp },
419             \fp_eval:n { \l__trt_sp_c_fp + \l__trt_sp_a_fp - \l__trt_sp_b_fp },
420             \fp_eval:n { \l__trt_sp_a_fp + \l__trt_sp_b_fp - \l__trt_sp_c_fp }
421             ) coordinate (#4);
422     \group_end:
423 }

```

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

4.4.9 X_9 – The *mittenpunkt*

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

```

424 \cs_new:Npn \trt_sp_mittenpunkt:nnnn #1 #2 #3 #4
425 {
426     \group_begin:
427         \__trt_bc_initialize:nnn {#1} {#2} {#3}
428         \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
429         \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
430         \path[overlay] (bc3 ~ cs \c_colon_str
431             \fp_eval:n
432             {

```

```

433         \l__trt_sp_a_fp * (
434             \l__trt_sp_b_fp + \l__trt_sp_c_fp - \l__trt_sp_a_fp
435         )
436     },
437     \fp_eval:n
438     {
439         \l__trt_sp_b_fp * (
440             \l__trt_sp_c_fp + \l__trt_sp_a_fp - \l__trt_sp_b_fp
441         )
442     },
443     \fp_eval:n
444     {
445         \l__trt_sp_c_fp * (
446             \l__trt_sp_a_fp + \l__trt_sp_b_fp - \l__trt_sp_c_fp
447         )
448     }
449 ) coordinate (#4);
450 \group_end:
451 }

```

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

4.4.10 X_{10} – The Spieker point

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

```

452 \cs_new:Npn \trt_sp_spieker:nnnn #1 #2 #3 #4
453 {
454     \group_begin:
455     \__trt_bc_initialize:nnn {#1} {#2} {#3}
456     \trt_distance_triangle:nnnNNN {#1} {#2} {#3}
457     \l__trt_sp_a_fp \l__trt_sp_b_fp \l__trt_sp_c_fp
458     \path[overlay] (bc3 ~ cs \c_colon_str
459         \fp_eval:n { \l__trt_sp_b_fp + \l__trt_sp_c_fp },
460         \fp_eval:n { \l__trt_sp_c_fp + \l__trt_sp_a_fp },
461         \fp_eval:n { \l__trt_sp_a_fp + \l__trt_sp_b_fp }
462     ) coordinate (#4);
463     \group_end:
464 }
465 </specialpoints>

```

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

4.5 The frontend layer

```

466 <frontend>
467 \ProvidesExplFile {trtfrontend.code.tex} {2020/04/30} {0.1}
468 {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`.

```

469 \tl_new:N \l__trt_fr_output_name_tl
470 \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.

```

471 \int_new:N \l__trt_fr_center_number_int

```

(End definition for `\l__trt_fr_center_number_int`.)

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

```
472 \tikzset {
473   triangletools/.is ~ family,
474   trt/.code={\pgfkeys{/tikz/triangletools/.cd,#1}},
475   triangletools/.cd,
```

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

`output_name` Change the output name of all returned coordinates.

```
476   output ~ name/.code={
477     \tl_set:Nn \l__trt_fr_output_name_tl {#1}
478   },
```

(End definition for `output name`. This function is documented on page 8.)

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

```
479   intersection/.code ~ args={(#1)(#2)--(#3)(#4)}{
480     \__trt_lt_return_intersection:nnnn {#1} {#2} {#3} {#4}
481     {\tl_use:N \l__trt_fr_output_name_tl}
482   },
483   foot ~ of ~ perpendicular/.code ~ args={(#1)--(#2)(#3)}{
484     \__trt_lt_return_perpendicular_coordinate:nnnn {#1} {#2} {#3}
485     {\tl_use:N \l__trt_fr_output_name_tl}
486   },
```

(End definition for `intersection` and `foot of perpendicular`. These functions are documented on page 5.)

`initialize_barycentric` The front-end of the barycentric coordinate system.

```
487   initialize ~ barycentric/.code ~ args={(#1)(#2)(#3)}{
488     \__trt_bc_initialize:nnn {#1} {#2} {#3}
489   },
```

(End definition for `initialize barycentric`. This function is documented on page 7.)

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

```
490   incenter/.code ~ args={(#1)(#2)(#3)}{
491     \trt_sp_incenter:nnnn {#1} {#2} {#3}
492     {\tl_use:N \l__trt_fr_output_name_tl}
493   },
494   excenter/.code ~ args={(#1)(#2)(#3)}{
495     \trt_sp_excenter:nnnn {#1} {#2} {#3}
496     {\tl_use:N \l__trt_fr_output_name_tl}
497   },
498   centroid/.code ~ args={(#1)(#2)(#3)}{
499     \trt_sp_centroid:nnnn {#1} {#2} {#3}
500     {\tl_use:N \l__trt_fr_output_name_tl}
501   },
502   circumcenter/.code ~ args={(#1)(#2)(#3)}{
503     \trt_sp_circumcenter:nnnn {#1} {#2} {#3}
504     {\tl_use:N \l__trt_fr_output_name_tl}
505   },
506   orthocenter/.code ~ args={(#1)(#2)(#3)}{
507     \trt_sp_orthocenter:nnnn {#1} {#2} {#3}
508     {\tl_use:N \l__trt_fr_output_name_tl}
509   },
```

(End definition for *incenter* and others. These functions are documented on page 2.)

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.

```

510 triangle ~ center/.code ~ args={(#1)(#2)(#3)(#4)}{
511   \int_case:nnF {#4}
512   {
513     {1} {
514       \trt_sp_incenter:nnnn {#1} {#2} {#3}
515       {\tl_use:N \l__trt_fr_output_name_tl}
516     }
517     {2} {
518       \trt_sp_centroid:nnnn {#1} {#2} {#3}
519       {\tl_use:N \l__trt_fr_output_name_tl}
520     }
521     {3} {
522       \trt_sp_circumcenter:nnnn {#1} {#2} {#3}
523       {\tl_use:N \l__trt_fr_output_name_tl}
524     }
525     {4} {
526       \trt_sp_orthocenter:nnnn {#1} {#2} {#3}
527       {\tl_use:N \l__trt_fr_output_name_tl}
528     }
529     {5} {
530       \trt_sp_ninepointcenter:nnnn {#1} {#2} {#3}
531       {\tl_use:N \l__trt_fr_output_name_tl}
532     }
533     {6} {
534       \trt_sp_symmedian:nnnn {#1} {#2} {#3}
535       {\tl_use:N \l__trt_fr_output_name_tl}
536     }
537     {7} {
538       \trt_sp_gergonne:nnnn {#1} {#2} {#3}
539       {\tl_use:N \l__trt_fr_output_name_tl}
540     }
541     {8} {
542       \trt_sp_nagel:nnnn {#1} {#2} {#3}
543       {\tl_use:N \l__trt_fr_output_name_tl}
544     }
545     {9} {
546       \trt_sp_mittenpunkt:nnnn {#1} {#2} {#3}
547       {\tl_use:N \l__trt_fr_output_name_tl}
548     }
549     {10} {
550       \trt_sp_spieker:nnnn {#1} {#2} {#3}
551       {\tl_use:N \l__trt_fr_output_name_tl}
552     }
553   }
554   {
555     \msg_error:nnn {triangletools} {center-not-found} {#4}
556   }
557 }
558 }
559 </frontend>

```

(End definition for *triangle center*. This function is documented on page 3.)

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.

Symbols		M	
/tikz/trt	2, 472	msg commands:	
B		\msg_error:nn	193
bc3	7, 186	\msg_error:nnn	555
bool commands:		\msg_new:nnnn	19 , 30 , 39
\bool_if:NTF	188	\msg_warning:nnnnnn	93
\bool_new:N	168	N	
\bool_set_true:N	172	\newdimen	7, 8
C		O	
centroid	3, 490	orthocenter	3, 490
circumcenter	3, 490	output_name	8, 476
\coordinate	118, 149 , 328 , 339 , 356 , 382	P	
cs commands:		\path	280 , 292 , 302 , 392 , 404 , 417 , 430 , 458
\cs_new:Npn	65, 88 , 103 , 114 , 121 , 131 , 145 , 170, 196 , 225 , 246 , 274 , 286 , 298 , 305, 360 , 369 , 386 , 398 , 411 , 424 , 452	\pgfextractx	67, 71 , 125 , 173 , 177 , 181, 227 , 231 , 308 , 312 , 316 , 374 , 378
E		\pgfextracty	68, 72 , 126 , 174 , 178 , 182, 228 , 232 , 309 , 313 , 317 , 375 , 379
excenter	2, 490	\pgfkeys	474
F		\pgfpointanchor	67, 68 , 71 , 72 , 125, 126 , 173 , 174 , 177 , 178 , 181 , 182, 227 , 228 , 231 , 232 , 308 , 309 , 312, 313 , 316 , 317 , 374 , 375 , 378 , 379
foot_of_perpendicular	5, 479	\ProvidesExplFile	17 , 50 , 154 , 219 , 254 , 467
fp commands:		\ProvidesExplPackage	5
\fp_compare:nNnTF	91	Q	
\fp_eval:n	281, 282 , 283 , 293 , 294 , 295 , 393, 394 , 395 , 405 , 406 , 407 , 418 , 419, 420 , 431 , 437 , 443 , 459 , 460 , 461	quark commands:	
\fp_new:N	52, 53 , 54 , 55 , 56 , 57 , 58 , 59, 60 , 61 , 62 , 63 , 64 , 159 , 160 , 161 , 162, 163 , 164 , 165 , 166 , 167 , 169 , 221, 222 , 223 , 224 , 256 , 257 , 258 , 259, 260 , 261 , 262 , 263 , 264 , 265 , 266, 267 , 268 , 269 , 270 , 271 , 272 , 273	\q_stop	190 , 196
\fp_set:Nn	69, 70 , 73 , 74, 75 , 79 , 83 , 90 , 95 , 96 , 99 , 100 , 123, 124 , 127 , 128 , 129 , 175 , 176 , 179, 180 , 183 , 184 , 198 , 199 , 200 , 201, 202 , 208 , 229 , 230 , 233 , 234 , 235, 310 , 311 , 314 , 315 , 318 , 319 , 320, 324 , 331 , 335 , 376 , 377 , 380 , 381	R	
\fp_to_dim:N	14, 119 , 150 , 214 , 215, 329 , 330 , 340 , 341 , 357 , 382 , 383	\RequirePackage	3, 4
G		S	
group commands:		str commands:	
\group_begin:	276, 288 , 300 , 307, 362 , 371 , 388 , 400 , 413 , 426 , 454	\c_colon_str	280, 292 , 302 , 392 , 404 , 417 , 430 , 458
\group_end:	284, 296 , 303 , 358, 367 , 384 , 396 , 409 , 422 , 450 , 463	T	
I		$\mathrm{T}_\mathrm{E}\mathrm{X}$ and $\mathrm{L}_\mathrm{T}_\mathrm{E}\mathrm{X} 2_\epsilon$ commands:	
incenter	2, 490	\pgfex	214
initialize_barycentric	7, 487	\pgfey	215
\input	9, 10 , 11 , 12 , 13 , 14	\trt@tmp@ei	7, 67 , 69 , 71 , 73 , 125, 127 , 173 , 175 , 177 , 179 , 181 , 183, 227 , 229 , 231 , 233 , 308 , 310 , 312, 314 , 316 , 318 , 374 , 376 , 378 , 380
int commands:		\trt@tmp@ii	7, 68 , 70 , 72 , 74 , 126, 128 , 174 , 176 , 178 , 180 , 182 , 184, 228 , 230 , 232 , 234 , 309 , 311 , 313, 315 , 317 , 319 , 375 , 377 , 379 , 381
\int_case:nnTF	511	\tikzdeclarecoordinatesystem	186
\int_new:N	471	\tikzset	472
intersection	5, 479	tl commands:	
M		\tl_new:N	469
N		\tl_set:Nn	470 , 477
O		\tl_use:N	481, 485 , 492, 496 , 500 , 504 , 508 , 515 , 519 , 523, 527 , 531 , 535 , 539 , 543 , 547 , 551
P		triangle_center	3, 510
Q		trt commands:	
R		\trt_distance:nnN	8, 8 , 225 , 248 , 249 , 250

\trt_distance_triangle:nnnNNN	8,	\l__trt_i_pointi_x_fp	69, 81, 85
246, 278, 290, 390, 402, 415, 428, 456		\l__trt_i_pointi_y_fp	70, 77, 85
\trt_lt_get_intersection_-		\l__trt_i_pointii_x_fp	73, 81
coordinate:nnnnNN	6, 103, 116	\l__trt_i_pointii_y_fp	74, 77
\trt_lt_get_intersection_-		\l__trt_lt_linei_a_fp	56, 106, 110, 134, 137, 141
line:NNNNNNNN	6, 6, 11, 88, 109, 140, 352	\l__trt_lt_linei_b_fp	56, 106, 110, 134, 137, 141
\trt_lt_get_line_equation:nnNNN	5, 65, 105, 107, 133, 342, 347	\l__trt_lt_linei_c_fp	56, 106, 110, 134, 137, 141
\trt_lt_get_perpendicular_-		\l__trt_lt_lineii_a_fp	56, 108, 111, 138, 142
coordinate:nnnNN	7, 12, 131, 147	\l__trt_lt_lineii_b_fp	56, 108, 111, 138, 142
\trt_lt_get_perpendicular_-		\l__trt_lt_lineii_c_fp	56, 108, 111, 138, 142
equation:nnNNNNN	6, 121, 136, 344, 349	\l__trt_lt_pointi_x_fp	52
\trt_sp_centroid:nnnn	3, 298, 499, 518	\l__trt_lt_pointi_y_fp	52
\trt_sp_circumcenter:nnnn	3, 305, 372, 503, 522	\l__trt_lt_pointii_x_fp	52
\trt_sp_excenter:nnnn	2, 286, 495	\l__trt_lt_pointii_y_fp	52
\trt_sp_gergonne:nnnn	4, 398, 538	\l__trt_lt_return_intersection:nnnnn	114, 365, 480
\trt_sp_incenter:nnnn	2, 274, 491, 514	\l__trt_lt_return_perpendicular_-	coordinate:nnnn 145, 363, 364, 484
\trt_sp_mittenpunkt:nnnn	4, 424, 546	\l__trt_lt_tmp_fp	11, 62, 90, 91, 99, 100
\trt_sp_nagel:nnnn	4, 411, 542	\l__trt_lt_tmpa_fp	62, 117, 119, 127, 129, 148, 150
\trt_sp_ninepointcenter:nnnn	3, 369, 530	\l__trt_lt_tmpb_fp	62, 117, 119, 128, 129, 148, 150
\trt_sp_orthocenter:nnnn	3, 360, 373, 507, 526	\l__trt_sp_a_fp	256, 279, 281, 291, 293, 391, 393, 403, 405, 406, 407, 416, 418, 419, 420, 429, 433, 434, 440, 446, 457, 460, 461
\trt_sp_spieker:nnnn	5, 452, 550	\l__trt_sp_b_fp	256, 279, 282, 291, 294, 391, 394, 403, 405, 406, 407, 416, 418, 419, 420, 429, 434, 439, 440, 446, 457, 459, 461
\trt_sp_symmedian:nnnn	4, 386, 534	\l__trt_sp_c_fp	256, 279, 283, 291, 295, 391, 395, 403, 405, 406, 407, 416, 418, 419, 420, 429, 434, 440, 445, 446, 457, 459, 460
trt internal commands:		\l__trt_sp_coordinatei_x_fp	259, 310, 322, 333
\l__trt_bc_anchor_iiix_fp	159, 183, 206	\l__trt_sp_coordinatei_y_fp	259, 311, 326, 337
\l__trt_bc_anchor_iiiy_fp	159, 184, 212	\l__trt_sp_coordinateii_x_fp	259, 314, 320, 322, 329
\l__trt_bc_anchor_iix_fp	159, 179, 205	\l__trt_sp_coordinateii_y_fp	259, 315, 324, 326, 330
\l__trt_bc_anchor_iiy_fp	159, 180, 211	\l__trt_sp_coordinateiii_x_fp	259, 318, 331, 333, 340
\l__trt_bc_anchor_ix_fp	159, 175, 204	\l__trt_sp_coordinateiii_y_fp	259, 319, 335, 337, 341
\l__trt_bc_anchor_iy_fp	159, 176, 210	\l__trt_sp_linei_a_fp	259, 346, 353
\l__trt_bc_initialize:nnn	14, 170, 277, 289, 301, 389, 401, 414, 427, 455, 488	\l__trt_sp_linei_b_fp	259, 346, 353
\l__trt_bc_initialized_bool	13, 168, 172, 188	\l__trt_sp_linei_c_fp	259, 346, 353
\l__trt_bc_lambda_i_fp	165, 199, 204, 210	\l__trt_sp_lineii_a_fp	259, 351, 354
\l__trt_bc_lambda_ii_fp	165, 200, 205, 211	\l__trt_sp_lineii_b_fp	259, 351, 354
\l__trt_bc_lambda_iii_fp	165, 201, 206, 212	\l__trt_sp_lineiii_c_fp	259, 351, 354
\l__trt_bc_parse:w	190, 196	\l__trt_sp_tmpa_fp	271, 343, 345, 348, 350, 355, 357, 376, 380, 382
\l__trt_bc_tmp_fp	169, 198, 199, 200, 201	\l__trt_sp_tmpb_fp	271, 343, 345, 348, 350, 355, 357, 377, 381, 383
\l__trt_d_pointi_x_fp	221, 229, 238, 239		
\l__trt_d_pointi_y_fp	221, 230, 241, 242		
\l__trt_d_pointii_x_fp	221, 233, 238, 239		
\l__trt_d_pointii_y_fp	221, 234, 241, 242		
\l__trt_fr_center_number_int	471		
\l__trt_fr_output_name_tl	469, 477, 481, 485, 492, 496, 500, 504, 508, 515, 519, 523, 527, 531, 535, 539, 543, 547, 551		

\l__trt_sp_tpc_fp	\l__trt_tmp_a_fp	202, 214
. <u>271</u> , 343, 345, 348, 350	\l__trt_tmp_b_fp	208, 215