Documentation for the geometry macros package `macros.mp`

Arseniy Akopyan
(akopjan@gmail.com)

Abstract

Some of these macros stolen from M. Vyaliy, I. Bogdanov, A. Polozov and V. Rodionov. The stealing is in progress.

Based command

Distance between $z0$ and $z1$:
```
abs(z0-z1);
```

Rotating of point $z1$ around $z0$ by an angle a:
```
z1 rotatedaround (z0, a);
```

Reflection of point $z0$ in a line passed through $z0$ and $z1$:
```
z0 reflectedabout(z1, z2);
```

Cut the path p by the segments $z0z1$ and $z1z2$:
```
p cutbefore (z0--z1) cutafter (z1--z2);
```

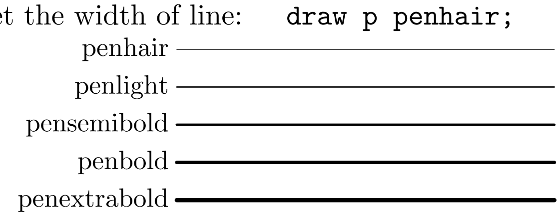
Length of the path p:
```
length p;
```

An addition shift of the letter:
```
dotlabel.adjusted.top(btex X etex, z0 adjust (0, 0.5));
```

Lines settings

Width

Set the width of line: `draw p penhair;`



Usually **penlight** is a default width of lines.

Labels and Marks

Label symbols

Label the point $z0$ by the symbol “X”:

```
label(btex $X$ etex, z0);
```

Draw the point $z0$ and label it by the symbol “X”:

```
dotlabel(btex $X$ etex, z0);
```

Label the point $z0$ by the symbol “X”, which is on white background:

```
whitelabel(btex $X$ etex, z0);
```

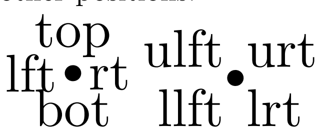
Draw the point $z0$ and label it by the symbol “X”, which is on white background:

```
whitedotlabel(btex $X$ etex, z0);
```

Draw the label under the marked point:

```
whitedotlabel.bot(btex $X$ etex, z0);
```

Another positions:



Marks

The radius of the arc in the angle mark: `angle_radius`. Size of signs in marks of paths `marksize`.

Square for right angle $z0z1z2$:

```
mark_rt_angle(z0, z1, z2);
```

If angle $z1$ is not right then it draw rhombus.

Dashed square for right angle $z0z1z2$:

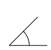
```
markdashed_rt_angle(z0, z1, z2);
```

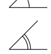
Square with a certain size for right angle $z0z1z2$:

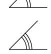
```
mark_rt_angle_withsize(z0, z1, z2, 10);
markdashed_rt_angle_withsize(z0, z1, z2, 10);
```

Last parameter is a size of the square. If it less than 0, then the macros use `angle_radius` for it.

Arcs for angle $z0z1z2$. The last parameter is the radius of arc, if it is non-positive then it will be equal `angle_radius`.

`arcs(z0, z1, z2, 10);` 

`arcs2(z0, z1, z2, 10);` 

`arcs3(z0, z1, z2, 10);` 

Same as `arcs()` with a label:

```
labelarcs(z0, z1, z2, 10, btex $\alpha$ etex);
```

Same as `labelarcs()` with a label on the white background:

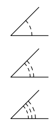
```
whitelabelarcs(z0, z1, z2, 10, btex $\alpha$ etex);
```

Same as `labelarcs()` and `whitelabelarcs()` with a certain distance from the vertex to the label:

```
labelarcsprof(z0, z1, z2, 10, 12, btex $\alpha$ etex);
whitelabelarcsprof(z0, z1, z2, 10, 12, btex $\alpha$ etex);
```

Dashed arcs for angle $z0z1z2$. The last parameter is the radius of arc, if it is non-positive then it will be equal `angle_radius`.

```
dashedarcs(z0, z1, z2, 10);
dashedarcs2(z0, z1, z2, 10);
dashedarcs3(z0, z1, z2, 10);
```



Same as `dashedarcs()` with a label: `labeldashedarcs(z0, z1, z2, 10, btex $\alpha$ etex);`

Same as `labeldashedarcs()` with a label on the white background:

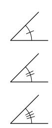
```
whitelabeldashedarcs(z0, z1, z2, 10, btex $\alpha$ etex);
```

Same as `labeldashedarcs()` and `whitelabeldashedarcs()` with a certain distance from the vertex to the label:

```
labeldashedarcsprof(z0, z1, z2, 10, 12, btex $\alpha$ etex);
whitelabeldashedarcsprof(z0, z1, z2, 10, 12, btex $\alpha$ etex);
```

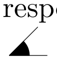
Marked arcs for angle $z0z1z2$. The last parameter is the radius of arc, if it is non-positive then it will be equal `angle_radius`.

```
mark_angle(z0, z1, z2, 10);
mark_angle2(z0, z1, z2, 10);
mark_angle3(z0, z1, z2, 10);
```



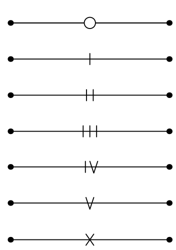
Filled segment for angle $z0z1z2$. The last parameter is the radius of arc, if it is non-positive then it will be equal `angle_radius`. For color responds global parameter `anglecolor`.

```
fill_angle(z0, z1, z2, 10);
```



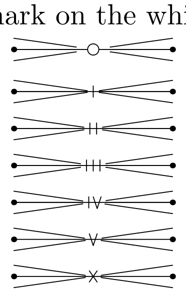
Mark the paths by roman numbers from 0 up to 5 and 10. Allows applying for more than one path (`rimmark(p1, p2)`). It does not draw the path..

```
rimmark0(p);
rimmark(p);
rimmark2(p);
rimmark3(p);
rimmark4(p);
rimmark5(p);
rimmark10(p);
```



Same as `rimmark()` with a mark on the white background:

```
whiterimmark0(p);
whiterimmark(p);
whiterimmark2(p);
whiterimmark3(p);
whiterimmark4(p);
whiterimmark5(p);
whiterimmark10(p);
```



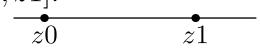
Path drawing

Draw the more than one path:

```
Draw p1, p2, p3;
```

Segment with the ends outside $[z0, z1]$:

```
ddline(z0, z1)(0.2, 0.4);
```



Same as `ddline()()` for paths:

```
ddrealline(p)(0.2, 0.4);
```

Arc with the end at $z0$ and $z2$ which passes through $z1$:

```
arc(z0, z1, z2);
```

Arc $z0z1z2$ which has ends outside $z0$ and $z2$:

```
ddarc(z0, z1, z2)(10, 20);
```

Parameters should be in degree.

Part of the circle p which lies between $z0$ and $z1$, (counter clockwise):

```
cutcircle(p, z0, z1);
```

Points labeling

Standard way to draw a point $z0$:

```
dotlabel("", z0);
```

Also you can use following macros. These macros also work with lists of points:

Circles and lines

All circles have length 4 (note that **fullcircle** has a length 8).

Circle with center at $z0$ and radius r:

```
circle(z0, r);
```

Radius could be a vector:

```
circle(z0, z1-z2);
```

Circle which pass through points $z0$, $z1$ and $z2$:

```
circumcircle(z0, z1, z2);
```

Intersection of lines $z0z1$ and $z2z3$:

```
crosspoint(z0, z1)(z2, z3);
```

Intersection of two curves (lines, circles and parabolas) $p0$ and $p1$:

```
cross(p0, p1);
```

Possible values of parameters: `.first`, `.second`, `.top`, `.bot`, `.lft` and `.rt`.

Tangent line to circle $p0$ from the point $z0$:

```
support(p0, z0);
```

It returns the point of tangent. Possible values of parameters: `.first`, `.second`, `.top`, `.bot`, `.lft` and `.rt`. Sometimes works with parabolas.

External common tangent to two circles $p0$ and $p1$.

```
dbltangent1(p0, p1);
dbltangent2(p0, p1);
dbltangent11(p0, p1);
dbltangent12(p0, p1);
dbltangent21(p0, p1);
dbltangent22(p0, p1);
```

First number respond number of tangent line, second one to the number of circle. If it has only one number then returns the path which connect two points of tangent.
If this tangent lines do not exist it will return an error message.

Internal common tangent to two circles $p0$ and $p1$.

```
dblintangent1(p0, p1);
dblintangent2(p0, p1);
dblintangent11(p0, p1);
dblintangent12(p0, p1);
dblintangent21(p0, p1);
dblintangent22(p0, p1);
```

First number respond number of tangent line, second one to the number of circle. If it has only one number then returns the path which connect two points of tangent.
If this tangent lines do not exist it will return an error message.

If point $z0$ is on a circle with center at $z2$. Than the second point of intersection of line $z0z1$ with a circle could be found by this macros:

```
secondpoint(z0, z1, z2);
```

Construct center of the circle inscribed in angle $z0z1z2$ and passed through the point $z3$:

```
angle_circle_in(z0, z1, z2, z3);
angle_circle_out(z0, z1, z2, z3);
```

Inversion with respect to the circle with center at $z0$, and radius r:

```
inversion(z0, r)(p);
```

p could be a point, a line or a circle.

Image of p under homothety with center at $z0$ and coefficient k:

```
scaleabout(z0, k)(p);
```

Conics and other curves

Ellipse with half-axis a and b and center at origin:

```
ellipse_canonical(a, b);
```

Ellipse with foci at $z.f1$ and $z.f2$ passes thought a point $z0$:

```
ellipseFFP(z.f1, z.f2, z0);
```

Ellipse with foci at $z.f1$ and $z.f2$ touches a line $z0z1$:

```
ellipseFFT(z.f1, z.f2, z0, z1);
```

Hyperbola with foci at $z.f1$ and $z.f2$ passes thought a point $z0$. Default length of hyperbola equals 88. It could regulate by `hyp_start` and `hyp_final`.

```
hyperbolaFFP(z.f1, z.f2, z0);
```

Hyperbola with foci at $z.f1$ and $z.f2$ touches a line $z0z1$. Default length of hyperbola equals 88. It could regulate by `hyp_start` and `hyp_final`.

```
hyperbolaFFT(z.f1, z.f2, z0, z1);
```

Arc of the hyperbola $xy = c^2$. Default length of hyperbola equals 88 and it's parameterized by polar angle (from 1 up to 89).

```
hyperbolaxy(c);
```

Left and right arcs of hyperbola with half-axis a and b and center at origin. Default length of hyperbola equals 88. It could regulate by `hyp_start` and `hyp_final`.

```
hyperbola_canonical_positive(a, b);
hyperbola_canonical_negative(a, b);
```

Parabola $y = ax^2 + bx + c$:

```
parabola_canonical(a, b, c)(e, l);
```

(e, l) coordinates of the points $(1, 1)$.

Parabola with focus at $z0$ and directrix $z1, z2$:

```
parabolaFD(z0, z1, z2);
```

Conics which passes through five points:

```
fivepointsconic(z0, z1, z2, z3, z4);
```

If the conic is hyperbola its return one of the arcs. The second arc is returned by command `fivepointsconic2(...)`.

Conics which touches five lines $z0z1$, $z2z3$, $z4z5$, $z6z7$ and $z8z9$:

```
inscribed_in_pentagon_conic(z0, z1)(z2, z3)(z4, z5)(z6,z7)(z8, z9);
```

If the conic is hyperbola its return one of the arcs. The second arc is returned by command `inscribed_in_pentagon_conic2(...)`.

Point with parameter k on a path p. It assumes, that the path has normal parametrization and its length equals 1.

```
pointonpath(p, k);
```

Elements of triangle

Base of bisector of the angle $z1$ in triangle $z0z1z2$:

```
bisector(z0, z1, z2);
```

Base of external bisector of the angle $z1$ in triangle $z0z1z2$:

```
exbisector(z0, z1, z2);
```

If base is far, then it return base of usual bisector rotated on $90°$ around $z1$.

Center of inscribed circle of triangle $z0z1z2$:

```
incenter(z0, z1, z2);
```

Center of escribed circle of triangle $z0z1z2$ which corresponds to the vertex $z0$:

```
excenter(z0, z1, z2);
```

Inscribed circle of triangle $z0z1z2$:

```
incircle(z0,z1,z2);
```

Escribed circle of triangle $z0z1z2$ which corresponds to the vertex $z0$:
```
excircle(z0,z1,z2);
```

Altitude of triangle $z0z1z2$ with the vertex at $z1$:
```
altitude(z0, z1, z2);
```

Orthocenter of triangle $z0z1z2$:
```
orthocenter(z0, z1, z2);
```

The median of triangle $z0z1z2$ with the vertex at $z1$:
```
median(z0, z1, z2);
```

Centroid of triangle $z0z1z2$:
```
centroid(z0, z1, z2);
```

Center of circumscribed circle of triangle $z0z1z2$:
```
circumcenter(z0, z1, z2);
```

Euler circle of triangle $z0z1z2$:
```
euler_circle(z0, z1, z2);
```

The isogonal conjugate point to a point $z3$ with respect to the triangle $z0z1z2$:
```
isogonal_point(z0, z1, z2)(z3);
```

The Nagel points of triangle $z0z1z2$:
```
nagel_point(z0, z1, z2);
```

The Gergonne points of triangle $z0z1z2$:
```
gergonne_point(z0, z1, z2);
```

The Lemoine points of triangle $z0z1z2$:
```
lemoine_point(z0, z1, z2);
```

The Torricelli points of triangle $z0z1z2$:
```
torricelli_point(z0, z1, z2);
torricelli_point2(z0, z1, z2);
```
Default point is the point which lies “inside” the triangle.

The Apollonius point of triangle $z0z1z2$:
```
apollonius_point(z0, z1, z2);
apollonius_point2(z0, z1, z2);
```
Default point is the point which lies “inside” the triangle.

The Brocard points of triangle $z0z1z2$:
```
brocard_point(z0, z1, z2);
brocard_point2(z0, z1, z2);
```

The Feuerbach point of triangle $z0z1z2$:
```
feuerbach_point(z0, z1, z2);
```

Pedal circle of point $z3$ with respect to triangle $z0z1z2$:
```
pedal_circle(z0, z1, z2)(z3);
```

Cevian circle of point $z3$ with respect to triangle $z0z1z2$:

```
cevian_circle(z0, z1, z2)(z3);
```

Inscribed in triangle $z0z1z2$ conic with prospector at $z3$:

```
inscribed_in_triangle_conic(z0, z1, z2)(z3);
```

Other curves

Lemniscate of Bernoulli with foci at $z0$ and $z1$:

```
lemniscate_of_bernoulli(z0, z1);
```

Length of the curve is equal 180. Parametrized by polar angle with center at the cusp.

Cardioid with cusp at $z0$ and the vertex at $z1$:

```
cardioid(z0, z1);
```

Length of the curve is equal 360. Parametrized by polar angle with center at the cusp.

Cissoid of Diocles with cusp at $z0$ and base of altitude to asymptote at $z1$:

```
dissoid_of_diocles(z0, z1);
```

Length of the curve is equal 120. Parametrized by polar angle with center at the cusp.