



The `hawkdraw` package

A LaTeX package to generate vector graphic output based on the `l3draw` package and related LaTeX3 modules using a simple syntax

Jasper Habicht *

Version 0.0.7, released on 25 May 2026

1 Introduction

The `hawkdraw` package allows the user to use a simple syntax similar to `TikZ` to generate vector graphic output based on `l3draw` and related LaTeX3 modules such as `l3fp` which provides very precise calculations. The syntax is meant to be simple and flexible but also coherent and logical with the aim of allowing the user to quickly draw graphics. The package name relates to the quickness and agility of hawks and the preciseness of their eyesight.

The package intends to define a set of user-level commands that can be used for drawing simple graphics. It does not aim to be a substitute for the `TikZ` package or to offer a similar scope of application.

The package is currently still in an experimental stage. The code may be subject to fundamental and frequent changes. The author is grateful for reporting any bugs via GitHub at <https://github.com/jasperhabicht/hawkdraw/issues>. A site for asking questions about the package and for suggestions for improvement is available at <https://github.com/jasperhabicht/hawkdraw/discussions>.

2 Loading the package

To install the package, copy the package file `hawkdraw.sty` into the working directory or into the `texmf` directory. After the package has been installed, the `hawkdraw` package is loaded by calling `\usepackage{hawkdraw}` in the preamble of the document.

The package can be used with PDFLaTeX, LuaLaTeX or XeLaTeX. The package does not load any dependencies, but it needs a LaTeX kernel of 1 June 2022 or newer. It is recommended to use the package with an up-to-date TeX distribution.

Note that the underlying code of the `l3draw` package is experimental as well and hence might as well change possibly breaking the functionality of this package.

* E-mail: mail@jasperhabicht.de. I am grateful to Joseph Wright, Max Chernoff and Clea F. Rees who helped me improving the code. Victor, the drawing hawk: © 2026 Hannah Klöber.

3 Main user environment

The package has only very few main user commands. Among these, the `hawkdraw` environment is the most important as it is used to define a drawing and most other commands are only defined inside this environment.

Similar to `TikZ`, the `hawkdraw` package makes use of key-value pairs to set options for commands. Some options are available for almost all commands, other have very specific uses. In the descriptions of the options, this manual states which data type the relevant option assumes. The following data types exist:

Data type	Explanation
<code>boolean</code>	A boolean only allows the values <code>true</code> or <code>false</code> . If not stated otherwise, setting the key without a value is equivalent with setting the key to <code>true</code> .
<code>string</code>	A string is a concatenation of letters or numbers of any length. Typically, strings do not contain symbols. For example, strings are used for names of points, frames of nodes, patterns or arrow tips.
<code>float</code>	A floating-point number is a number without or with unit that uses a colon as decimal separator.
<code>tuple</code>	A tuple consists of two floating-point numbers without or with unit that are separated by a comma. A tuple may be enclosed by parenthesis.
<code>dimension</code>	A dimension or length is a typical TeX length consisting of a floating point number and a length unit.
<code>clist</code>	A comma-separated list is a list of strings that are separated by commas. Such a list can consist of no or only one item as well. If the relevant list can only have a maximum number of items, this is stated in the description of the relevant key.
<code>options</code>	A key-value list consists of comma separated entities of keys that may be followed by an equals sign and a value. A value can be another key-value list.
<code>color</code>	This means a color definition as specified by the <code>l3color</code> module. Note that colors defined by the <code>xcolor</code> package have no effect when using the <code>hawkdraw</code> package.
<code>none</code>	Some keys do not accept any data type as they are only executing some function.

```
\begin{hawkdraw}[<options>]  
\end{hawkdraw}
```

The `hawkdraw` environment is the main environment for drawing. It defines most other commands described in the following sections. The following options can be set for the `hawkdraw` environment:

```
instance/baseline={<dimension>}
```

The option `baseline` sets the baseline for the environment as dimension. The default baseline is typically the bottom of the the bounding box of the drawing.

```
instance/layers={<clist>}
```

The option `layers` sets the layers for the environment as a comma-separated list. The layer `main` must always be present and is the default layer. Layers are drawn in the order they are defined (see [below](#)).

```
instance/overlay={<boolean>}
```

The option `overlay` sets the environment to have a zero-sized bounding box.



```
foo
\begin{hawkdraw}[overlay, baseline={0.5ex}]
  \stroke[stroke color={blue}]
    (0cm,0cm) circle[radius={0.5cm}] ;
\end{hawkdraw}
bar
```

4 Scopes and layers

```
\begin{scope}[<options>]
\end{scope}
```

The `scope` environment is only available inside the `hawkdraw` environment. It can be used to create scopes for the localization of path states, style settings and clipping regions.



```
\begin{hawkdraw}[fill color={blue}]
  \fill (0cm,0cm) circle[radius={5pt}] ;
  \begin{scope}[fill color={black}]
    \fill (1cm,0cm) circle[radius={5pt}] ;
  \end{scope}
  \fill (2cm,0cm) circle[radius={5pt}] ;
\end{hawkdraw}
```

```
scope/layer={<string>}
```

The option `layer` sets the scope to be placed on the relevant layer. The layer must be defined beforehand using the `layers` option.



```
\begin{hawkdraw}[layers={background,main}]
  \fill[fill color={blue}]
    (0cm,0cm) circle[radius={0.5cm}] ;
  \begin{scope}[layer={background}]
    \fill (0.5cm,0cm) circle[radius={0.5cm}] ;
  \end{scope}
\end{hawkdraw}
```

5 Path syntax

5.1 Path command and options

```
\path <path operations> ;
```

The command `\path` is used to draw a path inside the `hawkdraw` environment. The path is defined by a sequence of points and path construction operations. Each `\path` command must end with a semicolon. The following operations can be used to construct the path:

```
[<options>]
```

Options can be for the path using square brackets. The options are the same as for `\hawkdrawset`. Options can be set multiple times for the same path after any point on the path. Some options (such as color options or transformations) will apply to the entire path and the last specified value will be used.

5.2 Points on the path

Generally, the package makes use of the `l3fp` module of the LaTeX kernel. As such, it is possible to execute calculations on the fly to define the points. The basic form of a point is a tuple of two expressions resulting in a floating-point number where both expressions are separated by a comma.

It is advisable to enclose the two expressions of the tuple in curly braces in cases where complicated calculations should be executed in order to create a group to ensure proper evaluation and avoiding conflicts with parsing the coordinates, especially when using a syntax for polar points as described below.

```
(<float>,<float>)
```

Points on the path can be defined using x and y parts as floating-point numbers. The first item in the tuple is the x-coordinate, and the second item is the y-coordinate.

Points are per default defined as tuples representing the x- and y-coordinate. If floating-point numbers without unit are used, points are assumed per default. If other units are used, these are converted into points internally. The below alternative representations of points are as well calculated into the default tuple representation:

```
(<float>><float>)  
(<float>,<float>><float>)
```

Points on the path can also be defined using radius and angle parts as floating-point numbers. The first item in the tuple is the radius and the second item is the angle. The first item can contain a comma to define two radii.

```
(<float>:<float>)  
(<float>:<float>,<float>)
```

Points on the path can also be defined using radius and angle parts as floating-point numbers in reversed order. The first item in the tuple is the angle, and the second item is the radius. The second item can contain a comma to define two radii.

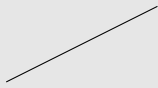
```
(<string>)
```

Points on the path can finally be referenced using names as strings. Points on the path can be named by either using the `point` or the `node` operation with the given name set via the `name` option (see [below](#)).

5.3 Path operations

```
-- (<point>)
```

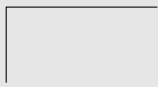
Using `--` a line to the next point is drawn.



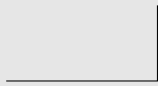
```
\begin{hawkdraw}
  \path[stroke] (0cm,0cm)
    -- (2cm,1cm) ;
\end{hawkdraw}
```

```
| - (<point>)
- | (<point>)
```

Using `| -` a vertical, then horizontal line to the next point is drawn. Using `- |` a vertical, then horizontal line to the next point is drawn.



```
\begin{hawkdraw}
  \path[stroke] (0cm,0cm)
    | - (2cm,1cm) ;
\end{hawkdraw}
```



```
\begin{hawkdraw}
  \path[stroke] (0cm,0cm)
    - | (2cm,1cm) ;
\end{hawkdraw}
```

```
.. (<control point>) .. (<point>)
.. (<control point>) & (<control point>) .. (<point>)
```

The above syntax draws a curve to the next point with a single control point or with two control points.



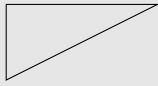
```
\begin{hawkdraw}
  \path[stroke] (0cm,0cm)
    .. (1cm,1cm)
    .. (2cm,1cm);
\end{hawkdraw}
```



```
\begin{hawkdraw}
  \path[stroke] (0cm,0cm)
    .. (0cm,0.5cm) & (1cm,1cm)
    .. (2cm,1cm);
\end{hawkdraw}
```

```
!
```

Using `!` the path is closed by drawing a line from the last point to the first point.



```
\begin{hawkdraw}
\path[stroke] (0cm,0cm)
-- (2cm,1cm)
-- (0cm,1cm) ! ;
\end{hawkdraw}
```

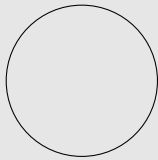
The following path operations require at least one option to be set:

circle[<options>]

Using the **circle** operation, a circle is drawn with the current point as center.

path/circle/**radius**={<dimension>}

The radius is specified as a dimension with the **radius** option.



```
\begin{hawkdraw}
\path[stroke] (0cm,0cm)
circle[radius={1cm}] ;
\end{hawkdraw}
```

ellipse[<options>]

Using the **ellipse** operation, an ellipse is drawn with the current point as center.

```
path/ellipse/radius a={<dimension>}
path/ellipse/radius b={<dimension>}
path/ellipse/vector a={<tuple>}
path/ellipse/vector b={<tuple>}
```

The vectors of the ellipse are specified as tuples of dimensions with the **vector a** and **vector b** options. Alternatively, the options **radius a** and **radius b** can be used to define the radii in the x- and y-dimension.



```
\begin{hawkdraw}
\path[stroke] (0cm,0cm)
ellipse[
radius a={0.5cm},
radius b={1cm}
] ;
\end{hawkdraw}
```




```
\begin{hawkdraw}
\path[stroke] (0cm,0cm)
ellipse[
vector a={(0.5cm,1cm)},
vector b={(0.75cm,0cm)}
] ;
\end{hawkdraw}
```

arc[<options>]


Using the **arc** operation, an arc is drawn with the current point as origin.

```
path/arc/vector a={<tuple>}
path/arc/vector b={<tuple>}
path/arc/radius a={<dimension>}
path/arc/radius b={<dimension>}
path/arc/angle start={<float>}
path/arc/angle end={<float>}
path/arc/angle delta={<float>}
```


The vectors of the arc are specified as tuples of dimensions with the **vector a** and **vector b** options. Alternatively, the options **radius a** and **radius b** can be used to define the radii in the x- and y-dimension. The angles are specified as dimensions with the **start angle** and **end angle** options. Instead of the **angle end** option, the option **angle delta** can be used.



```
\begin{hawkdraw}
\path[stroke] (0cm,0cm)
  arc[
    radius={1cm},
    angle start={45},
    angle end={135}
  ] ;
\end{hawkdraw}
```



```
\begin{hawkdraw}
\path[stroke] (0cm,0cm)
  arc[
    radius a={0.5cm},
    radius b={1cm},
    angle start={45},
    angle delta={90}
  ] ;
\end{hawkdraw}
```



```
\begin{hawkdraw}
\path[stroke] (0cm,0cm)
  arc[
    vector a={(0.5cm,1cm)},
    vector b={(0.75cm,0cm)},
    angle start={45},
    angle end={135}
  ] ;
\end{hawkdraw}
```

rectangle[<options>]

Using the **rectangle** operation, an rectangle with the current point as origin corner is drawn.

```
path/rectangle/corner={<tuple>}  
path/rectangle/relative={<boolean>}
```

The other corner of the rectangle is specified as a point (a tuple) with the **corner** option. If the boolean option **relative** is set, the corner point is interpreted as relative to the current point instead of absolute.



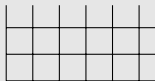
```
\begin{hawkdraw}  
  \path[stroke] (0cm,0cm)  
    rectangle[corner={(2cm,1cm)}] ;  
\end{hawkdraw}
```

grid[<options>]

Using the **grid** operation, a grid with the current point as origin corner is drawn.

```
path/grid/step={<tuple>}  
path/grid/corner={<tuple>}
```

The step is specified as a comma-separated list of at most two dimensions with the **step** option. The first item in the tuple is the step in the x-direction, and the second item is the step in the y-direction. If only one dimension is given, it is used for both directions. The other corner is specified as a point (a tuple) with the **corner** option.



```
\begin{hawkdraw}  
  \path[stroke] (0cm,0cm)  
    grid[step={10pt}, corner={(2cm,1cm)}] ;  
\end{hawkdraw}
```



```
\begin{hawkdraw}  
  \path[stroke] (0cm,0cm)  
    grid[step={5pt,10pt}, corner={(2cm,1cm)}] ;  
\end{hawkdraw}
```

5.4 Points and nodes

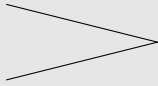
point[<options>]

Using the **point** operation, a point at the current position is defined and named with the given name.

```
path/point/at={<tuple>}  
path/point/name={<string>}
```

The position of the point can be adjusted with the **at** option which accepts a point (a tuple). The point can be referenced later by using the name in parentheses (see [above](#)).

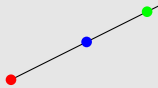
The command `\point` is a shortcut for `\path (opt, opt) point`



```
\begin{hawkdraw}
  \path (0,0) point[name=A, at={(2cm,0.5cm)}} ;
  \path[stroke] (0cm,0cm) -- (A) -- (0cm,1cm) ;
\end{hawkdraw}
```

`path/point/at path part={⟨float⟩}`

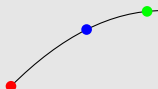
The position of a point can also be adjusted using the `at path part` option which accepts a floating-point number that defines the position on the last path segment with 0 being the previous and 1 being the current point.



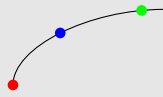
```
\begin{hawkdraw}
  \path[stroke] (0cm,0cm)
    -- (2cm,1cm)
    point[name=A, at path part={0}]
    point[name=B, at path part={0.5}]
    point[name=C, at path part={0.9}] ;
  \fill[fill color=red] (A) circle[radius=2pt] ;
  \fill[fill color=blue] (B) circle[radius=2pt] ;
  \fill[fill color=green] (C) circle[radius=2pt] ;
\end{hawkdraw}
```



```
\begin{hawkdraw}
  \path[stroke] (0cm,0cm)
    -| (2cm,1cm)
    point[name=A, at path part={0}]
    point[name=B, at path part={0.5}]
    point[name=C, at path part={0.9}] ;
  \fill[fill color=red] (A) circle[radius=2pt] ;
  \fill[fill color=blue] (B) circle[radius=2pt] ;
  \fill[fill color=green] (C) circle[radius=2pt] ;
\end{hawkdraw}
```



```
\begin{hawkdraw}
  \path[stroke] (0cm,0cm)
    .. (1cm,1cm)
    .. (2cm,1cm)
    point[name=A, at path part={0}]
    point[name=B, at path part={0.5}]
    point[name=C, at path part={0.9}] ;
  \fill[fill color=red] (A) circle[radius=2pt] ;
  \fill[fill color=blue] (B) circle[radius=2pt] ;
  \fill[fill color=green] (C) circle[radius=2pt] ;
\end{hawkdraw}
```



```
\begin{hawkdraw}
\path[stroke] (0cm,0cm)
  .. (0cm,0.5cm) & (1cm,1cm)
  .. (2cm,1cm)
point[name={A}, at path part={0}]
point[name={B}, at path part={0.5}]
point[name={C}, at path part={0.9}] ;
\fill[fill color=red] (A) circle[radius=2pt] ;
\fill[fill color=blue] (B) circle[radius=2pt] ;
\fill[fill color=green] (C) circle[radius=2pt] ;
\end{hawkdraw}
```



```
\begin{hawkdraw}
\path[stroke] (0cm,0cm)
  arc[
    radius={1cm},
    angle start={45},
    angle end={135}
  ]
point[name={A}, at path part={0}]
point[name={B}, at path part={0.5}]
point[name={C}, at path part={0.9}] ;
\fill[fill color=red] (A) circle[radius=2pt] ;
\fill[fill color=blue] (B) circle[radius=2pt] ;
\fill[fill color=green] (C) circle[radius=2pt] ;
\end{hawkdraw}
```



```
\begin{hawkdraw}
\path[stroke] (0cm,0cm)
  arc[
    vector a={(0.5cm,1cm)},
    vector b={(0.75cm,0cm)},
    angle start={45},
    angle end={135}
  ]
point[name={A}, at path part={0}]
point[name={B}, at path part={0.5}]
point[name={C}, at path part={0.9}] ;
\fill[fill color=red] (A) circle[radius=2pt] ;
\fill[fill color=blue] (B) circle[radius=2pt] ;
\fill[fill color=green] (C) circle[radius=2pt] ;
\end{hawkdraw}
```

node[<options>]{<content>}

Using the `node` operation, a node at the current position is defined and assigned the relevant content by the relevant argument.

```

path/node/at={<tuple>}
path/node/name={<string>}
path/node/at path part={<float>}
path/node/anchor={<clist>}

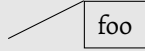
```

The position of the node can be adjusted with the `at` option which accepts a point (a tuple) or with the `at path part` which accepts a floating-point number (see [above](#)). The anchor is specified with the `anchor` option which accepts a comma-separated list of at most two poles that define the anchor point as their intersection (see the explanations on coffins). Available poles are `t`, `b`, `l`, `r`, `vc` and `hc`. The node can be named by setting the `name` option.

```

foo          \begin{hawkdraw}
              \node {foo} ;
              \end{hawkdraw}

```



```

\begin{hawkdraw}
\stroke (0cm,0cm) -- (1cm,0.5cm)
node[anchor={t,l}, stroke] {foo} ;
\end{hawkdraw}

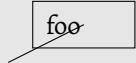
```

```

path/node/width={<dimension>}

```

The width of the node can be set with the `width` option.



```

\begin{hawkdraw}
\stroke (0cm,0cm) -- (1cm,0.5cm)
node[stroke, width={1cm}] {foo} ;
\end{hawkdraw}

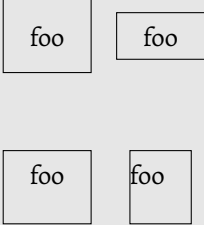
```

```

path/node/padding={<clist>}

```

Using the `padding` option, the padding between the border of the node and the contents is set. The value is a comma-separated list of one to four dimensions. If four dimensions are given, they apply to the top, left, bottom and right side of the node respectively. If three dimensions are given, the first applies to the top, the second to the left and right side, the third to the bottom. If two dimensions are given, the first applies to the top and bottom and the second to the left and right sides. If only one dimension is given, it is applied to all four sides.



```

\begin{hawkdraw}
  \node[
    stroke,
    at={(0cm,2cm)},
    padding={10pt}
  ] {foo} ;
  \node[
    stroke,
    at={(1.5cm,2cm)},
    padding={5pt,10pt}
  ] {foo} ;
  \node[
    stroke,
    at={(0cm,0cm)},
    padding={5pt,10pt,15pt}
  ] {foo} ;
  \node[
    stroke,
    at={(1.5cm,0cm)},
    padding={5pt,10pt,15pt,0pt}
  ] {foo} ;
\end{hawkdraw>


```

```
path/node/text color={<color>}
path/node/text opacity={<float>}
```

Using the `text color` option, the color of the node contents is set. The default text color is the current (text) color. The option `text opacity` sets the text opacity. The floating-point number should be in the range of 0 and 1. The default text opacity is 1 (fully opaque). Note that `\DocumentMetadata{}` needs to be set in order to activate support for PDF transparency.

```
path/node/frame={<string>}
```

Using the `frame` command, the frame surrounding the node text can be selected. The frame `rectangle` is set per default. All options that are available for paths are also available for node frames.






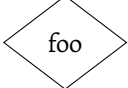
```

\begin{hawkdraw}
  \stroke (0cm,0cm) -- (1cm,0.5cm)
  node[
    fill, fill color={yellow},
    text color={blue},
    frame={ellipse}
  ] {foo} ;
\end{hawkdraw>

```

The command `\node` is a shortcut for `\path (opt,opt) node`
The following node frames are available, the frame `none` is special as it removes the bounding box from the node altogether:

Node frame	Name
foo	none

	rectangle
	ellipse
	circle
	diamond

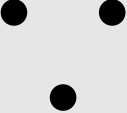
6 Mapping operation

map[<options>]{<code>}

The **map** operation is used to map over a comma-separated list of items or a range of numbers and execute code for each item that is given by the relevant argument. The code can reference the current item with **#1**. The following options can be set for the command:

```
path/map/list={<clist>}
path/map/start={<float>}
path/map/end={<float>}
path/map/step={<float>}
```

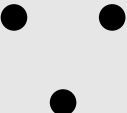
The option **list** sets the command to map over a comma-separated list of items. Each item in the list is passed as an argument to the code.



```
\begin{hawkdraw}
\fill
  map[list={30,150,270}] {
    (#1:0.75cm) circle[radius={5pt}]
  } ;
\end{hawkdraw}
```

The option **start** sets the start of the range for mapping over a range of numbers as a floating-point number. The default start is 1. The option **end** sets the end of the range for mapping over a range of numbers as a floating-point number. The default end is 1. The option **step** sets the step for the range for mapping over a range of numbers as a floating-point number. The default step is 1.

The command **\map** is a shortcut for **\path (opt,opt) map**



```
\begin{hawkdraw}
\map[start={30}, step={120}, end={270}] { [fill]
  (#1:0.75cm) circle[radius={5pt}]
} ;
\end{hawkdraw}
```

7 Main path options

\hawkdrawset{<options>}

The command `\hawkdrawset` is used to set options for paths. It can be used inside or outside of `hawkdraw` environments.

The following options can be set for the path:

7.1 Stroke and fill color, clipping

```
path/stroke
path/fill
path/clip
```

These options add a stroke to the path, fill the path, or use the path as a clipping region for subsequent paths. They can be used in combination. As the clipping region affects all following paths, it is often useful to use it in combination with scopes (see [above](#)).

The command `\stroke` is a shortcut for `\path [stroke]`.

The command `\fill` is a shortcut for `\path [fill]`

The command `\clip` is a shortcut for `\path [clip]`



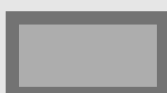
```
\begin{hawkdraw}
  \path[stroke, clip] (0cm,0cm)
    rectangle[corner={(2cm,1cm)}} ;
  \fill (1cm,0.5cm)
    circle[radius={0.6cm}] ;
\end{hawkdraw}
```

```
path/stroke color={<color>}
path/fill color={<color>}
path/stroke opacity={<float>}
path/fill opacity={<float>}
```

The options `stroke color` and `fill color` set the stroke color and the fill color for the path. The default stroke color and fill color is the current (text) color.



```
\begin{hawkdraw}[line width={5pt}]
  \path[
    fill, fill color={yellow},
    stroke, stroke color={blue}
  ] (0cm,0cm)
    rectangle[corner={(2cm,1cm)}} ;
\end{hawkdraw}
```



```
\begin{hawkdraw}[line width={5pt}]
  \path[
    fill, fill opacity={0.25},
    stroke, stroke opacity={0.5}
  ] (0cm,0cm)
    rectangle[corner={(2cm,1cm)}} ;
\end{hawkdraw}
```

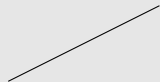
The options `stroke opacity` and `fill opacity` set the stroke opacity and the fill opacity for the path. The floating-point number should be in the range of 0 and 1. The default stroke opacity

and fill opacity is 1 (fully opaque). Note that `\DocumentMetadata{}` needs to be set in order to activate support for PDF transparency.


7.2 Line styling

```
path/line width={<dimension>}
```

Sets the line width for the path. The default line width is 0.4 pt.




```
\begin{hawkdraw}
\stroke
(0cm,0cm) -- (2cm,1cm);
\end{hawkdraw}
```



```
\begin{hawkdraw}
\stroke[line width={5pt}]
(0cm,0cm) -- (2cm,1cm) ;
\end{hawkdraw}
```

```
path/line cap={<string>}
path/line join={<string>}
path/miter limit={<float>}
```


The option `line cap` sets the line cap for the path. The possible values are `butt`, `round`, and `rectangle`. The default line cap is `butt`.



```
\begin{hawkdraw}[line width={5pt}]
\stroke[line cap={round}]
(0cm,0cm) -- (2cm,1cm) ;
\end{hawkdraw}
```

The option `line join` sets the line join for the path. The possible values are `miter`, `round`, and `bevel`. The default line join is `miter`.

The option `miter limit` sets the miter limit for the path. The default miter limit is 10.

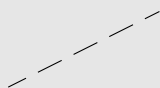


```
\begin{hawkdraw}[line width={5pt}]
\stroke[line join={bevel}]
(0cm,0cm) -- (2cm,0.5cm)
-- (0cm,1cm) ;
\end{hawkdraw}
```

```
path/dash pattern={<clist>}
path/dash phase={<dimension>}
```

The option `dash pattern` sets the dash pattern for the path as a comma-separated list of dimensions. Each item in the list is a dimension representing the length of a dash or a gap. The default dash pattern is empty (a solid line).

The option `dash phase` sets the dash phase for the path as a dimension. The default dash phase is 0 pt.



```
\begin{hawkdraw}
  \stroke[
    dash pattern={10pt,5pt},
    dash phase={2.5pt}
  ] (0cm,0cm) -- (2cm,1cm) ;
\end{hawkdraw}
```

7.3 Rounded corners and full rule

path/**corner arc**={<clist>}

The option **corner arc** the corner arc for the path as a comma-separated list of at most two dimensions. The first applies to the corner formed by the path leading into the corner, and the second applies to the corner formed by the path leading out of the corner. If only one dimension is given, it applies to both corners. The default corner arc is 0 pt (sharp corners).



```
\begin{hawkdraw}
  \stroke[corner arc={5pt}] (0cm,0cm)
    rectangle[corner={(2cm,1cm)}] ;
\end{hawkdraw}
```

path/**fill rule**={<string>}

The option **fill rule** sets the fill rule for the path. The possible values are **non-zero** and **even-odd**. The default fill rule is **non-zero**. Note that this option is not influenced by scoping.

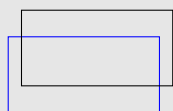


```
\begin{hawkdraw}[fill rule={even-odd}]
  \fill (0cm,0cm)
    rectangle[corner={(2cm,1cm)}]
    (1cm,0.5cm)
    circle[radius={0.4cm}] ;
\end{hawkdraw}
```

7.4 Transformations

path/**shift**={<tuple>}
 path/**rotate**={<float>}
 path/**scale**={<tuple>}
 path/**slant**={<tuple>}

The option **shift** sets the shift for the path as a tuple of dimensions. The first item in the tuple is the shift in the x-direction, and the second item is the shift in the y-direction. The default shift is (0 pt, 0 pt).



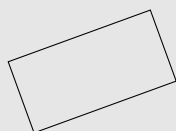
```
\begin{hawkdraw}
  \stroke[stroke color={blue}]
    (0cm,0cm) rectangle[corner={(2cm,1cm)}}] ;
  \stroke[shift={(5pt,10pt)}]
    (0cm,0cm) rectangle[corner={(2cm,1cm)}}] ;
\end{hawkdraw}
```

The option **rotate** sets the rotation for the path as a floating-point number representing the angle of rotation. The default rotation is 0° .



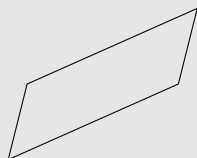
```
\begin{hawkdraw}
  \stroke[scale={0.25,0.5}]
    (0cm,0cm) rectangle[corner={(2cm,1cm)}}] ;
\end{hawkdraw}
```

The option **scale** sets the scale for the path as a comma-separated list of at most two floating-point numbers. The first item in the tuple is the scale in the x-direction, and the second item is the scale in the y-direction. If only one number is given, it is used for both directions. The default scale is (1, 1).



```
\begin{hawkdraw}
  \stroke[rotate={20}]
    (0cm,0cm) rectangle[corner={(2cm,1cm)}}] ;
\end{hawkdraw}
```

The option **slant** sets the slant for the path as a comma-separated list of at most two floating-point numbers. The first item in the tuple is the slant in the x-direction, and the second item is the slant in the y-direction. If only one number is given, it is used for both directions. The default slant is (0, 0).



```
\begin{hawkdraw}
  \stroke[slant={0.25,0.5}]
    (0cm,0cm) rectangle[corner={(2cm,1cm)}}] ;
\end{hawkdraw}
```

8 Setting styles and colors

```
path/style set={<options>}
path/style add={<options>}
```

To simplify styling, custom keys can be defined via the keys **style set** and **style add**. Both keys accept as value a key-value list consisting of one or multiple custom keys whose relevant value consists of another key-value list describing the relevant style. It is possible to use **#1** as placeholder for one argument. It is also possible to reference to an already define custom key in the definition of another custom key.

The key **style set** defines custom keys and sets their values. The key **style add** appends the relevant key-value list to the existing custom key as defined by **style set**. Note that it is not checked whether a custom key already exists and its definition will be overwritten without a warning.



```
\begin{hawkdraw}[
  style set={
    mystyle={
      stroke, stroke color={blue},
      line width={5pt}
    }
  }
]
  \path[mystyle]
    (0cm,0cm) -- (2cm,1cm) ;
\end{hawkdraw}
```

\hawkdrawsetcolor{<color>}{<model>}{<value>}

The command `\hawkdrawsetcolor` is used to define colors using the `l3color` module. It is recommended to define custom colors in the preamble of the document. If the model is set to `named`, the command expects as value a color expression.

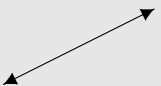


```
\hawkdrawsetcolor{colorA}{rgb}{0.1,0.5,0.8}
\hawkdrawsetcolor{colorB}{named}{yellow!90!red}
\begin{hawkdraw}[line width={5pt}]
  \path[
    stroke, stroke color={colorA},
    fill, fill color={colorB}
  ] (0cm,0cm) rectangle[corner={(2cm,1cm)}] ;
\end{hawkdraw}
```

8.1 Arrow tips

`path/arrow start`={<string>}
`path/arrow end`={<string>}

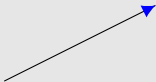
The options `arrow start` and `arrow end` set an arrow tip to the start and the end of the current path respectively. Both keys take as value a string representing the name of the relevant arrow tip. The `default` arrow tip is selected per default. Arrow tips are not attached to single points or closed paths.



```
\begin{hawkdraw}
  \stroke[arrow start, arrow end]
    (0cm,0cm) -- (2cm,1cm) ;
\end{hawkdraw}
```








`path/arrow style set`={<options>}
`path/arrow style add`={<options>}
`path/arrow start style set`={<options>}
`path/arrow start style add`={<options>}
`path/arrow end style set`={<options>}
`path/arrow end style add`={<options>}

Arrow tips can be styled using the options `arrow start style set` and `arrow end style set`. The option `arrow style set` sets the same style for start and end arrow tips. The options `arrow start style add` and `arrow style add` append the relevant styles to the existing styles. All options that are available for paths are also available for arrow tips.



```
\begin{hawkdraw}
  \stroke[
    arrow end,
    arrow style set={fill, fill color=blue}
  ]
  (0cm,0cm) -- (2cm,1cm) ;
\end{hawkdraw}
```

The following arrow tips are available, of these the last three, `simple`, `straight` and `bar`, should be set together with the option `arrow style set={stroke}`:

Arrow tip	Name
	<code>none</code>
	<code>default</code>
	<code>stealth</code>
	<code>triangle</code>
	<code>circle</code>
	<code>simple</code>
	<code>straight</code>
	<code>bar</code>

8.2 Patterns

Note that in order to be able to use patterns, `\DocumentMetadata{}` needs to be set.

```
path/fill pattern={<string>}
```

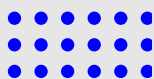
The option `fill pattern` sets a fill pattern for the current path. The key takes as value a string representing the name of the relevant pattern.

```
path/pattern style set={<options>}
path/pattern style add={<options>}
```

Patterns can be styled using the options `pattern style set` and `pattern style add`. The option `pattern style set` sets the style for the pattern of the current path. The options `pattern style add` appends the relevant styles to the existing styles. All options that are available for paths are also available for patterns.


```
path/pattern x={<dimension>}
path/pattern y={<dimension>}
```

The options `pattern x` and `pattern y` set the dimensions of the pattern in the x and y directions, respectively. The default dimensions of a pattern in both directions are both 10 pt.



```
\begin{hawkdraw}
  \path[
    fill pattern={dots},
    pattern style set={fill color=blue}
  ]
    (0cm,0cm) rectangle[corner={(2cm,1cm)}] ;
\end{hawkdraw}
```

The following patterns are available:

Pattern	Name
	
	lines
	hatch
	dots

9 User commands

\firstpoint
\lastpoint

The commands `\firstpoint` and `\lastpoint` can be used to access the first and last point of the current path, respectively.

\firstslope
\lastslope

The commands `\firstslope` and `\lastslope` can be used to access the slope at the first and last point of the current path, respectively. The slope is represented as the angle with respect to the positive x-axis.

10 Changes

v0.0.5 (2026/05/18) First public alpha release.

v0.0.6 (2026/05/20) Splitting up code into modules.

v0.0.7 (2026/05/25) Adding patterns; adding `at path part` option.