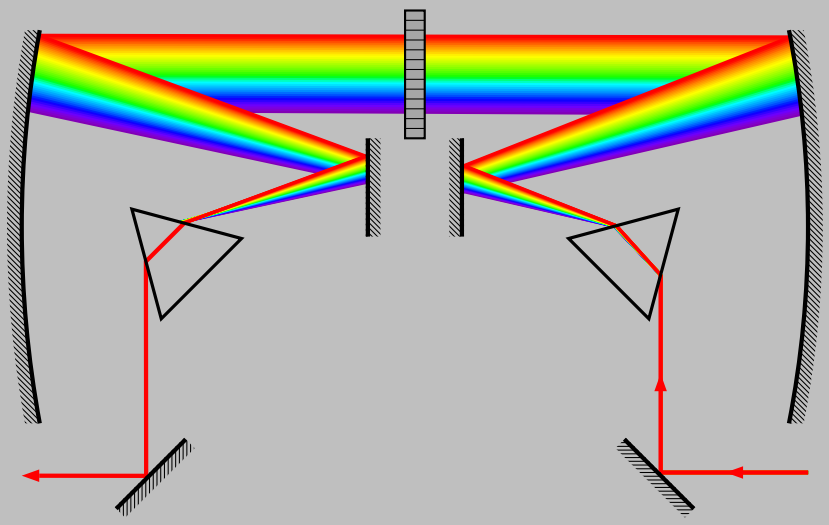
PSTricks

pst-optexp

Drawing optical experimental setups

v3.6

2013/03/20



Package author:
Christoph Bersch

Contents

1. Introduction

1.1. About the package

The package `pst-optexp` is a collection of optical components that facilitate easy sketching of optical experimental setups. A lot of different free-ray and fiber components are provided, which alignment, positioning and labelling can be achieved in very simple and flexible ways. The components can be connected with fibers or beams, realistic raytraced beam paths are also possible.

1.2. Requirements

`pst-optexp` requires LaTeX and recent versions of `pst-node`, `multido`, `pstricks-add`, `pst-eucl`, and `environ`.

All PSTricks package rely heavily on the Postscript language so that the typical workflow involves `latex`, `dvips`, and `ps2pdf`. Of course there are several alternative ways to compile your documents.[1]

1.3. Distribution and installation

This package is available on CTAN[2] and is included in TeXLive and MiKTeX.

The `pst-optexp` package consists of the two main files `pst-optexp.ins` and `pst-optexp.dtx`. By running `tex pst-optexp.ins` the following derived files are generated:

- `pst-optexp.pro`: the Postscript prolog file
- `pst-optexp.sty`: the LaTeX style file

[1] http://tug.org/PSTricks/main.cgi?file=pdf/pdfoutput
[2] http://mirror.ctan.org/help/Catalogue/entries/pst-optexp.html

Save the files in a directory which is part of your local TEX tree.

Do not forget to run `texhash` to update this tree. For MiKTEX users, do not forget to update the file name database (FNDB).

For more detailed information see the documentation of your personal LATEX distribution on installing packages to your local TEX system.

1.4. License

Permission is granted to copy, distribute and/or modify this software under the terms of the LATEX Project Public License, version 1.3c.[3] This package is author-maintained.

1.5. Backward compatibility

Version 3.0 introduced a lot of advanced features which made it too difficult to maintain full backward compatibility with version 2.1. Especially the `\drawbeam` macro was broken by design, so that it was best to drop the old flaw as early as possible.

You find a more detailled list of changes and information for migration of old documents in Sec. 11.3.

1.6. Notation

Often the parameter types are self-explanatory, but in some cases a clear distinction is needed, e.g. between ⟨*num*⟩ and ⟨*psnum*⟩. Tab. 1.1 explains some commonly used types.

[3] http://www.latex-project.org/lppl.txt

Name	description
num	float number
psnum	Postscript code which evaluates to a number
int	integer number
dimen	dimension
psstyle	custom graphics parameter configuration defined with `\newpsstyle`
refpoint	Like the reference point of `\rput`, can be any comination of `c` (center), `t` (top), `b` (bottom), `l` (left), and `r` (right).

Table 1.1.: Parameter types used in the parameter reference.

1.7. Acknowledgements

I thank all the people of the PSTricks mailinglist for the continuous help, especially Herbert Voß. Thanks also to various package authors from which I learned and adopted code for this package, Florent Chervet, Rolf Niepraschk and Heiko Oberdiek. Christine Römer convinced me with her article in the german DTK[4] to provide a german translation of the documentation. The documentation style is a mixture of the `pst-doc` class (Herbert Voß) and the `ltxdockit` package for the `biblatex` documentation (Philipp Lehman).

[4] Pakete in Deutsch dokumentieren. in Die TeXnische Komödie. Heft 2/2011, S. 28-35.

2. Basic ideas

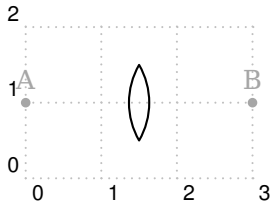
This chapter shows the basic ideas and concepts of this package. Starting with simple examples, basic functionality such as alignment, positioning (2.1) and labelling (2.2) of components is presented. Following, the connection of components with beam rays (2.3) or fiber (2.4) is demonstrated.

In Sec. 2.5 several propositions for preparing large experimental sketches are illustrated step by step. For a complete reference of all macros and their parameters, see Sec. 3–9.

A drawing consists of components, which can be positioned, shifted and rotated based on their reference nodes, and may have an optional label. Then, the components are connected with beam rays or fibers.

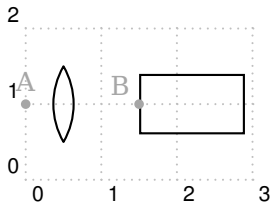
2.1. The components

A component is positioned according to the nodes which are used for its definition—the reference nodes. In the following examples, the lens is placed centered between nodes $\langle A \rangle$ and $\langle B \rangle$:




```
\begin{pspicture}[showgrid](0,-0.3)(3,2.3)
  \pnode(0,1){A}\pnode(3,1){B}
  \lens(A)(B)
\end{pspicture}
```

This position can be changed with a number of parameters, such as `abspos` (see Sec. 3.2 and Sec. 3.3). In the next example, the lens is positioned `0.5` units apart from node $\langle A \rangle$, the box is placed at the end of the reference line $\overline{AB}$.

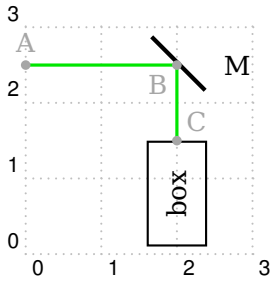



```
\begin{pspicture}[showgrid](0,-0.3)(3,2.3)
  \pnode(0,1){A}\pnode(1.5,1){B}
  \lens[abspos=0.5](A)(B)
  \optbox[position=end](A)(B)
\end{pspicture}
```

The components can be divided in two categories: free-ray and fiber components. These differ only in their connection possibilities: Free-ray components can have fiber or free-ray connections, whereas fiber components support fiber connections only.

2.2. Labels

Every component can have an optional label, which is positioned relative to the component.



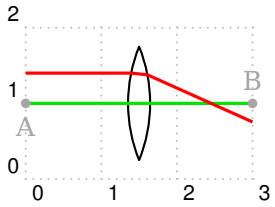

```
\begin{pspicture}[showgrid](0,-0.3)(3,3.3)
  \pnode(0,2.5){A}\pnode(2,2.5){B}\pnode(2,1.5){C}
  \mirror[labelangle=-45](A)(B)(C){M}
  \optbox[position=start, labeloffset=0,
      labelref=relative](C)(B){box}
  \drawbeam(A){1}{2}
\end{pspicture}
```

2.3. Free-ray connections

Free-ray components can be connected with beam rays, which path can be calculated by raytracing. Besides the realistic tracing possibilities, there are many ways to manipulate the beam paths because you are dealing with a package for *sketching* experimental setups.

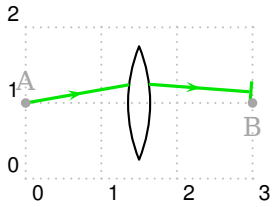
In the simplest case a single line is drawn, which optical path is determined by the refractive index `n` and Snell's law.

The example shows two beams which hit the lens with different initial conditions. The red beam starts at a distance of `0.2` from the optical axis (`beampos`) and is deflected by the lens accordingly.



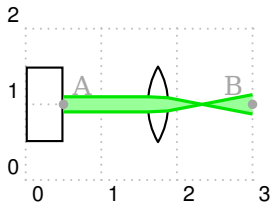

```
\begin{pspicture}[showgrid](0,-0.3)(3,2.3)
  \pnode(0,1){A}\pnode(3,1){B}
  \lens[n=2, lensradius=2 2, lensheight=1.5](A)(B)
  \drawbeam(A){1}(B)
  \drawbeam[beampos=0.4, linecolor=red](A){1}(B)
\end{pspicture}
```

The beams can be decorated with arbitrary dashs and arrows.



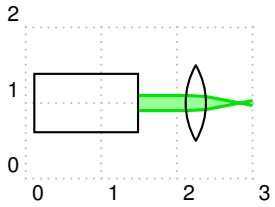

```
\begin{pspicture}[showgrid](0,-0.3)(3,2.3)
  \pnode(0,1){A}\pnode(3,1){B}
  \lens[n=2, lens=2 2 1.5](A)(B)
  \drawbeam[beamangle=10, ArrowInside=->, arrows=-|, ↩
      beaminside=false](A){1}(B)
\end{pspicture}
```

You can also draw wide beams. The marginal rays can be changed with `linestyle` etc. and the filling with `fillstyle` etc.




```
\begin{pspicture}[showgrid](0,-0.3)(3,2.3)
  \pnode(0.5,1){A}\pnode(3,1){B}
  \optbox[position=start, optboxsize=0.5 1](A)(B)
  \lens[n=2](A)(B)
  \drawwidebeam[beamwidth=0.2, fillstyle=solid, ↩
      fillcolor=green!40]{1-2}(B)
\end{pspicture}
```

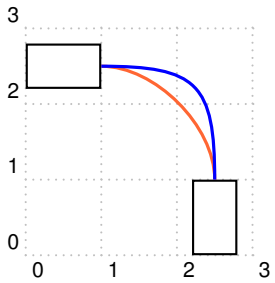
The beams can also lay behind the components. For this you must enclose all respective components and rays in a `optexp` environment (compare the previous example with the following one).




```
\begin{pspicture}[showgrid](0,-0.3)(3,2.3)
  \pnode(1.5,1){A}\pnode(3,1){B}
  \begin{optexp}
    \optbox[position=start](A)(B)
    \lens[n=2](A)(B)
    \drawwidebeam[beamwidth=0.2, fillstyle=solid, ↩
        fillcolor=green!40]{1}{2}(B)
  \end{optexp}
\end{pspicture}
```

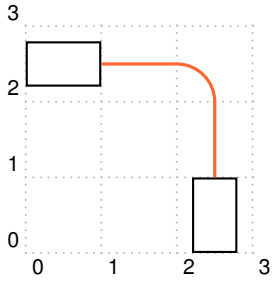
2.4. Fiber connections

Fibers (\drawfiber) are another possibility of connecting components. Here, you have several way of automatically adapting the fiber connection to the component alignment.




```
\begin{pspicture}[showgrid](0,-0.3)(3,3.3)
  \pnode(1,2.5){A}\pnode(2.5,1){B}
  \psset{optboxsize=1 0.6}
  \optbox[position=start](A)(B|A)
  \optbox[position=end](B|A)(B)
  \drawfiber{1}{2}
  \drawfiber[linecolor=blue, ncurv=1.2]{1}{2}
\end{pspicture}
```

Usually, the fibers are drawn as \nccurve connections, other node connections are possible as well (fiberstyle).



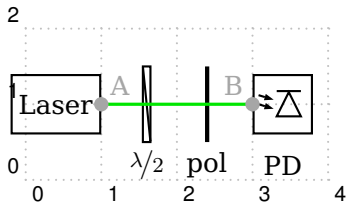

```
\begin{pspicture}[showgrid](0,-0.3)(3,3.3)
  \pnode(1,2.5){A}\pnode(2.5,1){B}
  \psset{optboxsize=1 0.6}
  \optbox[position=start](A)(B|A)
  \optbox[position=end](B|A)(B)
  \drawfiber[fiberstyle=angle, linearc=0.5]{1}{2}
\end{pspicture}
```

2.5. Step-by-step examples

In this first example, all components are distributed on a single line and connected with a beam.

First we define the start node ⟨A⟩ and the stop node ⟨B⟩. The first \optbox is positioned at the start, the wave plate and the polarizer are positioned between the nodes. The parameter position expects a value between 0 and 1 which is relative to the length of the reference line, or start/end. The detector is always positioned at the end of its reference line.

Finally, all components are connected with a simple beam ray.



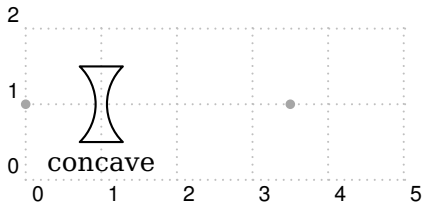

```
\begin{pspicture}[showgrid](4,2)
  \pnode(1,1){A}\pnode(3,1){B}
  \optbox[position=start, optboxwidth=1.2,
      labeloffset=0](A)(B){Laser}
  \optretplate[position=0.3](A)(B){$\nicefrac{\lambda}{2}$}
  \optplate[position=0.7](A)(B){pol}
  \optdetector[dettype=diode](A)(B){PD}
  \drawbeam{1}{2}{3}{4}
\end{pspicture}
```

2.5.1. Galilean telescope

This example describes a simple setup with a concave and a convex lens and the raytracing through them.

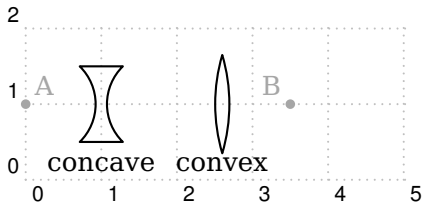
First, we define the reference nodes ⟨A⟩ and ⟨B⟩ which are used for alignment of the components. For better overview, these nodes are highlighted and a grid is drawn.

The first lens should have concave interfaces. With lensradius the interface radii are set to negative values, which correspond to concave curvatures. With abspos the lens is positioned one unit from ⟨A⟩ node.



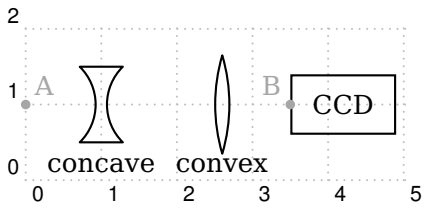

```
\begin{pspicture}[showgrid](0,-0.3)(5,2.3)
  \pnode(0,1){A}\pnode(3.5,1){B}
  \lens[lensradius=-0.7,
        abspos=1](A)(B){concave}
\end{pspicture}
```

The second lens has a convex curvature (lensradius is positive) and is positioned 2.6 units apart from ⟨A⟩. Parameter lensheight sets the total height of the lens.



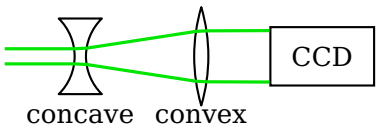

```
\begin{pspicture}[showgrid](0,-0.3)(5,2.3)
  \pnode(0,1){A}\pnode(3.5,1){B}
  \lens[lensradius=-0.7,
        abspos=1](A)(B){concave}
  \lens[lensradius=2.3, lensheight=1.3,
        abspos=2.6](A)(B){convex}
\end{pspicture}
```

The beam rays are supposed to end at the CCD camera which is drawn as a \optbox. With position=end the box is positioned at the end of the reference line from ⟨A⟩ to ⟨B⟩. The label is put at the center of the component with labeloffset.



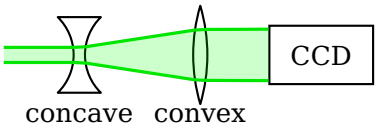

```
\begin{pspicture}[showgrid](0,-0.3)(5,2.3)
  \pnode(0,1){A}\pnode(3.5,1){B}
  \lens[lensradius=-0.7,
        abspos=1](A)(B){concave}
  \lens[lensradius=2.3, lensheight=1.3,
        abspos=2.6](A)(B){convex}
  \optbox[position=end, labeloffset=0](A)(B){CCD}
\end{pspicture}
```

Finally, a wide beam with an initial beam width of 0.2 without divergence is sent through the components. The beam path is calculated by raytracing. The lens parameters in this example were set such, that for a collimated input beam also the output beam is collimated.




```
\begin{pspicture}(5,2)
  \pnode(0,1){A}\pnode(3.5,1){B}
  \lens[lensradius=-0.7,
        abspos=1](A)(B){concave}
  \lens[lensradius=2.3, lensheight=1.3,
        abspos=2.6](A)(B){convex}
  \optbox[position=end, labeloffset=0](A)(B){CCD}
  \drawwidebeam[beamwidth=0.2](A){1}{2}{3}
\end{pspicture}
```

The appearance of the beam can be changed via the Beam style, and may have e.g. a semitransparent fill pattern.



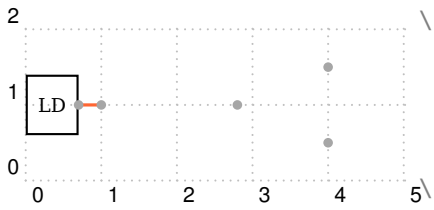

```
\begin{pspicture}(5,2)
  \pnode(0,1){A}\pnode(3.5,1){B}
  \lens[lensradius=-0.7,
        abspos=1](A)(B){concave}
  \lens[lensradius=2.3, lensheight=1.3,
        abspos=2.6](A)(B){convex}
  \optbox[position=end, labeloffset=0](A)(B){CCD}
  \addtopsstyle{Beam}{fillstyle=solid,
    fillcolor=green, opacity=0.2}
  \drawwidebeam[beamwidth=0.2](A){1}{2}{3}
\end{pspicture}
```

2.5.2. Fiber setup

This example shows how to setup a fiber drawing with automatic fiber connections.

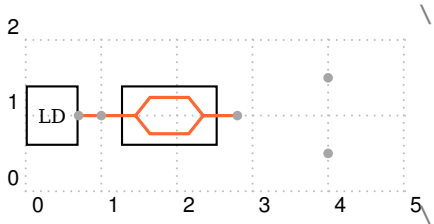
Again, the reference nodes are shown for better orientation. In contrast to the previous example, here, we need more reference nodes because the components are connected directly with them.

The laser is set at the start. The label offset is set with the `label` parameter, which would allow setting all label parameters at once. A `\optbox` is originally a free-ray component, so that for it the fiber connection must be enabled with `fiber`. The width of all `\optbox` is changed to `0.7`.



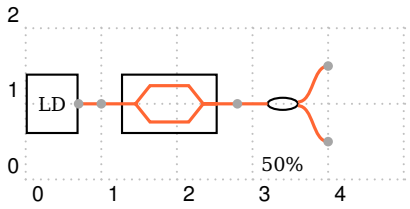

```
\begin{pspicture}[showgrid](4.7,2)
  \pnode(0.7,1){A}\pnode(1,1){B}\pnode(2.8,1){C}
  \pnode(4,1.5){D}\pnode(4,0.5){E}
  \psset{optboxwidth=0.7, labelstyle=\scriptsize}
  \optbox[position=start, label=0, fiber](A)(B){LD}
\end{pspicture}
```

The Mach–Zehner modulator is drawn without label and connected automatically with fibers to its reference nodes.




```
\begin{pspicture}[showgrid](4.7,2)
  \pnode(0.7,1){A}\pnode(1,1){B}\pnode(2.8,1){C}
  \pnode(4,1.5){D}\pnode(4,0.5){E}
  \psset{optboxwidth=0.7, labelstyle=\scriptsize}
  \optbox[position=start, label=0, fiber](A)(B){LD}
  \optmzm(B)(C)
\end{pspicture}
```

Then, we draw a fiber coupler with one input and two output fibers. The coupler is positioned between the input node and the center of the two output nodes.

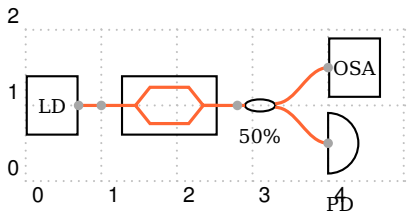



```
\begin{pspicture}[showgrid](4.7,2)
  \pnode(0.7,1){A}\pnode(1,1){B}\pnode(2.8,1){C}
  \pnode(4,1.5){D}\pnode(4,0.5){E}
  \psset{optboxwidth=0.7, labelstyle=\scriptsize}
  \optbox[position=start, label=0, fiber](A)(B){LD}
  \optmzm(B)(C)
  \wdmsplitter(C)(D)(E){50\%}
\end{pspicture}
```

At the ends of the coupler we put an optical spectrum analyzer (OSA) and a power meter (PD). The `\optdetector` is always set at the end of the reference line, it has `position` fixed to `end`. These two components are drawn without fibers, because their end nodes coincide with the two outputs of the coupler.

The notation `(A|E)` uses the x-coordinate of $\langle A \rangle$ and the y-coordinate of $\langle E \rangle$.

Ex. 2.1




```
\begin{pspicture}[showgrid](4.7,2)
  \pnode(0.7,1){A}\pnode(1,1){B}\pnode(2.8,1){C}
  \pnode(4,1.5){D}\pnode(4,0.5){E}
  \psset{optboxwidth=0.7, labelstyle=\scriptsize}
  \optbox[position=start, label=0, fiber](A)(B){LD}
  \optmzm(B)(C)
  \wdmsplitter[abspos=0.3, label=0.4](C)(D)(E){50\%}
  \optbox[position=end, label=0](A|D)(D){OSA}
  \optdetector(A|E)(E){PD}
\end{pspicture}
```

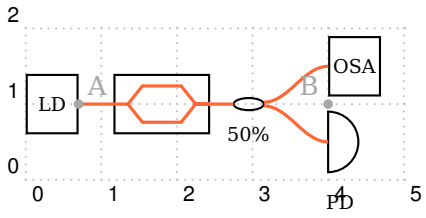
2.5.3. Fiber setup (alternative)

This example has the same result as the previous one (Ex. 2.5.2), but is constructed differently.

This time we do not use the automatic fiber connections, but draw all components first and connect them afterwards with fibers (`\drawfiber`). This requires less reference nodes, the fiber connections are suppressed globally with `fiber=none`.

The choice of the procedure depends on personal preferences, for loop setups, like for fiber lasers, it looks better if the fibers are drawn after the components with `\drawfiber`.

Ex. 2.2

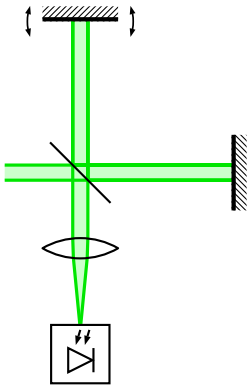



```
\begin{pspicture}[showgrid](4.9,2)
  \pnode(0.7,1){A}\pnode(4,1){B}
  \psset{optboxwidth=0.7, labelstyle=\scriptsize,
      fiber=none}
  \optbox[position=start, labeloffset=0](A)(B){LD}
  \optmzm[abspos=1.1](A)(B)
  \wdmsplitter[abspos=2.25, label=0.4](A)(B)(B){50\%}
  \optbox[position=end,label=0,
      compshift=0.5](A)(B){OSA}
  \optdetector[compshift=-0.5](A)(B){PD}
  \drawfiber{1}{2}{3}{4}
  \drawfiber{3}{5}
\end{pspicture}
```

2.5.4. Michelson-Interferometer

In this example we use lenses, reflective elements (mirrors) and components which can be either transmittive or reflective (beamsplitter).

Ex. 2.3



```
1  \begin{pspicture}(3.2,5)
2    \pnode(0,3){A}\pnode(1,3){BS}\pnode(3,3){M1}
3    \pnode(1,5){M2}\pnode(1,1){PD}
4    \psset{mirrortype=extended, mirrordepth=0.2}
5    \begin{optexp}
6      \beamsplitter[bsstyle=plate, compname=BS](A)(BS)(PD)
7      \mirror[compname=M1](BS)(M1)(BS)
8      \mirror[compname=M2, variable](BS)(M2)(BS)
9      \lens[compname=L](BS)(PD)
10     \optdetector[compname=Det, dettype=diode](BS)(PD)
11     \addtopsstyle{Beam}{beamwidth=0.2, fillstyle=solid,
           fillcolor=green!20!white}
12     \drawwidebeam(A){BS}{M1}{BS}{M2}{BS}{L}{Det}
13   \end{optexp}
14 \end{pspicture}
```

We start with definition of the reference nodes (Line 2–3) and setting some basic parameters (Line 4).

Following, all components are positioned (Line 6–10). The reflective ones need three reference nodes, the input node, the node where the reflective interface is situated, and the output node. Then, the appearance of the beam is set (Line 11), and the beam is drawn (Line 12).

In this example, all components have a name (`compname`), which can be used instead of the numerical IDs—automatically assigned, unique numbers—for the raytracing. This can simplify follwing the beam path.

All components and beams are enclosed in a `optexp` environment. Inside of this, all beams are drawn behind the components.

3. General component parameters

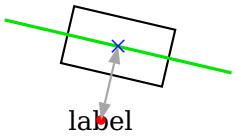
This chapter introduces generic parameters and concepts such as labels (3.1), positioning (3.2), rotating and shifting of components (3.3) and changing the appearance of components (3.4 and 3.5).

3.1. Labels

All components may have a label which position and alignment can be adjusted at will. The label is optional for every component. If the various parameters are not sufficient, you can access the label node—marked as red dot in some examples—directly (see also Sec. 7.4).

`labeloffset`=⟨*num*⟩ default: `0.8`

The offset of the label reference node (red dot) from the component center (blue cross).



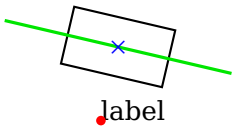

```
\begin{pspicture}(3,1.8)
  \optbox[beam, labeloffset=1](0,1.5)(3,0.8){label}
\end{pspicture}
```

`labelstyle`=⟨*macros*⟩

The text style that is used to typeset the label.

`labelalign`=⟨*refpoint*⟩ default: `c`

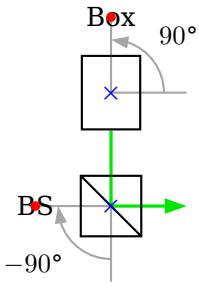
Defines the alignment of the label relative to the label node.




```
\begin{pspicture}(3,1.8)
  \psset{beam, labeloffset=1}
  \optbox[labelalign=bl](0,1.5)(3,0.8){label}
\end{pspicture}
```

labelangle=⟨*num*⟩ default: 0

The rotation angle of the label node around the component center. The origin depends both on the component and the choosen reference system (see labelref).

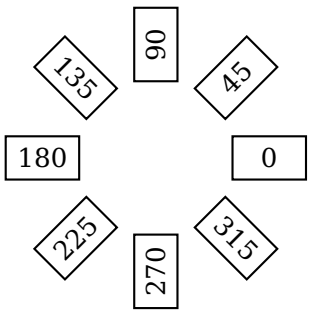



```
\begin{pspicture}(-0.3,0)(2,3.7)
  \psset{labeloffset=1, optboxwidth=1, arrowscale=1.5, arrowinset=0}
  \optbox[position=end, labelangle=90](1,1)(1,2){Box}
  \beamsplitter[labelangle=-90](1,2)(1,1)(2,1){BS}
  \drawbeam[arrows=->]{1}{2}(2,1)
\end{pspicture}
```

labelref=relative, relgrav, global, absolute default: relgrav

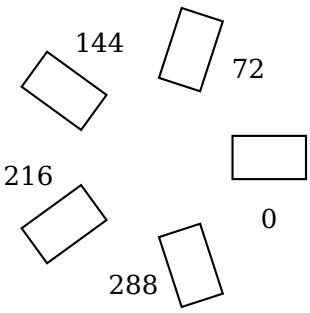
Set the reference coordinate system for the labelangle and the orientation of the label text.

relative The label is parallel to the component reference line and the actual angle is always projected to the range between $-90°$ and $+90°$.



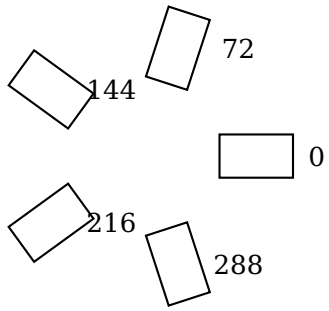

```
\begin{pspicture}(0,-2.2)(2,2.2)
  \psset{position=end, optboxsize=1 0.6, labeloffset=0,
      labelref=relative}
  \multido{\i=0+45}{8}{%
    \optbox(0,0)(1;\i){\i}
  }
\end{pspicture}
```

relgrav The label itself is always horizontal, only the label node is rotated together with the component.




```
\begin{pspicture}(0,-2.2)(2,2.2)
  \psset{position=end, optboxsize=1 0.6, labelref=relgrav}
  \multido{\i=0+72}{5}{%
    \optbox(0,0)(1;\i){\i}
  }
\end{pspicture}
```

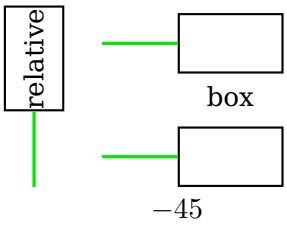
`global, absolute` The label is independent of the component alignment.




```
\begin{pspicture}(0,-2.2)(2,2.2)
  \psset{position=end, optboxsize=1 0.6, labelref=global}
  \multido{\i=0+72}{5}{%
    \optbox(0,0)(1;\i){\i}
  }
\end{pspicture}
```

`label`=⟨*offset*⟩[⟨*angle*⟩[⟨*refpoint*⟩[⟨*labelref*⟩]]]

Allows compact definition of several label parameters. It takes up to four space-separated arguments (`labeloffset`, `labelangle`, `labelalign`, and `labelref`). Unchanged intermediate arguments can be skipped with a dot.




```
\begin{pspicture}(0,0)(4,3)
  \psset{endbox, beam}
  \optbox[label=0.7](1.5,2.5)(2.5,2.5){box}
  \optbox[label=1 -45](1.5,1)(2.5,1){$-45$}
  \optbox[label=0 . . relative]%
         (0.6,0.6)(0.6,1.6){relative}
\end{pspicture}
```

`innerlabel=true`

This is an alias for `label=0 . . relative`

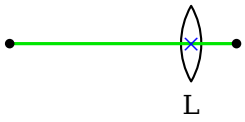
3.2. Positioning

All components except for the free-ray tripoles can be positioned between their two reference nodes `\oenodeRefA` and `\oenodeRefB`.

`position`=⟨*num*⟩, `start`, `end`

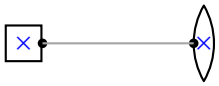
Controls the relative position of an object center (blue cross) between its two reference points (black dots), if the value is a number. In this case it is equivalent to the `npos` parameter of `\ncput`, i.e. a number in the range $[0, 1]$ (see `pst-node` documentation[1]).

[1] http://mirror.ctan.org/help/Catalogue/entries/pst-node.html




```
\begin{pspicture}(3,1.5)
  \lens[beam, position=0.8](0,1)(3,1){L}
\end{pspicture}
```

The values `start` and `end` place an object at the start or end of the reference line, respectively. The respective object interface is placed on the respective reference node (black dot) instead of the object center (blue cross) which would be the case for `position=0` or `position=1`. Achieving this with numerical values would me more difficult and unflexible, because the object size and shape must be taken into account.



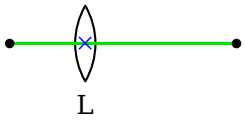
```
\begin{pspicture}(3,2)
  \pnode(0.5,1){A}\pnode(2.5,1){B}\psdot(A)\psdot(B)
  \optbox[optboxsize=0.5 0.5, position=start](A)(B)
  \lens[position=end](A)(B)
\end{pspicture}
```

Here is for comparison the positioning with `position=0/1`.

```
\begin{pspicture}(3,2)
  \pnode(0.5,1){A}\pnode(2.5,1){B}\psdot(A)\psdot(B)
  \optbox[optboxsize=0.5 0.5, position=0](A)(B)
  \lens[position=1](A)(B)
\end{pspicture}
```

abspos=⟨*num*⟩, `start`, `end`

Controls the absolute position of an object on the reference line. The values `start` and `end` are handled identically to position.




```
\begin{pspicture}(3,1.5)
  \lens[beam, abspos=1](0,1)(3,1){L}
\end{pspicture}
```

endbox=`true`, `false`

This parameter is equivalent to position=end.

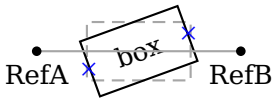
3.3. Rotating and shifting

All components can be shifted and rotated around different reference nodes.

Please note, that together with the components also their interface nodes (Sec. 7.6, marked as blue crosses in the following examples) are transformed, which may impact the connections. The transformed reference nodes are accessible as new nodes (Sec. 7.2).

`angle`=⟨*num*⟩ default: `0`

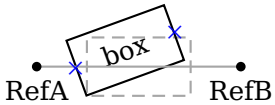
Rotate a component by an angle ⟨*num*⟩ (in degree). The original position is drawn gray dashed.




```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(2.7,1){B}
  \optbox[angle=20, innerlabel](A)(B){box}
\end{pspicture}
```

`rotateref`=⟨*refpoint*⟩ default: `c`

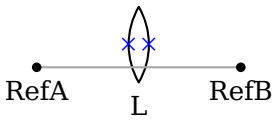
Set the reference point for the rotation of the component. Please see Sec. 7.7 for a detailed explanation and Sec. 11.2.1 for a list of the possible reference nodes of all components. The original position is drawn gray dashed.



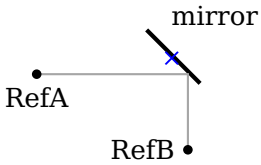

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(2.7,1){B}
  \optbox[angle=20, rotateref=bl, innerlabel](A)(B){box}
\end{pspicture}
```

`compshift`=⟨*num*⟩ default: `0`

Shift a component perpendicular to its reference line (see Sec. 7.2). For reflective components it is shifted along the reflective interface.



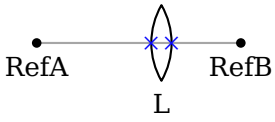

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(2.7,1){B}
  \lens[compshift=0.3](A)(B){L}
\end{pspicture}
```



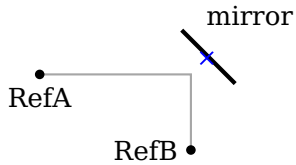

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(2,0){B}
  \mirror[compshift=0.3](A)(2,1)(B){mirror}
\end{pspicture}
```

`compoffset`=⟨*num*⟩ default: `0`

Shift a component perpendicular to its reference line (see Sec. 7.2). For reflective components it is shifted along the reflective interface. The displacement is vertical to a displacement with `compshift`. For an example see Ex. 10.



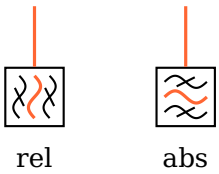

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(2.7,1){B}
  \lens[compoffset=0.3](A)(B){L}
\end{pspicture}
```




```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(2,0){B}
  \mirror[compoffset=0.3](A)(2,1)(B){mirror}
\end{pspicture}
```

`innercompalign`=`rel`, `relative`, `abs`, `absolute`

If this parameter is set to `absolute`, the "inner" component part is not rotated with the component. At the moment, this option is used only by `\optfilter`.



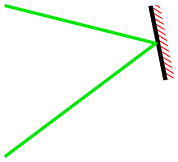

```
\begin{pspicture}(3,2)
\psset[optexp]{labelangle=-90, position=start}
\optfilter[innercompalign=rel](0.5,1.2)(0.5,2){rel}
\optfilter[innercompalign=abs](2.5,1.2)(2.5,2){abs}
\end{pspicture}
```

3.4. Using psstyles

Psstyles are custom graphics parameter configurations for PSTricks objects. The `pst-optexp` package makes extensive use of styles to facilitate flexible design of single, logical parts of components. This is in general not possible with the optional argument.

Existing styles can be extended with `\addtopsstyle` and overwritten with `\newpsstyle`.

As example we consider an `extended` mirror and want to change only the "extended" part which is controlable with the `ExtendedMirror` style:



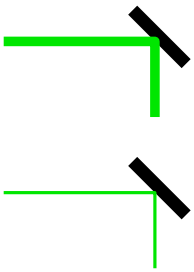
```
\begin{pspicture}(3,2)
  \addtopsstyle{ExtendedMirror}{hatchcolor=red}
  \mirror[mirrortype=extended, beam](0,0)(2,1.5)(0,2)
\end{pspicture}
```

The changeable parts of each component are listed in the respective reference.

3.5. Component appearance

OptComp ⟨*psstyle*⟩

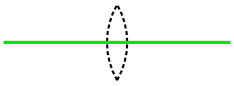
Affects the basic appearence of all optical components. Using standard graphics parameters like e.g. `linestyle` would change also the connections that are drawn together with the component (`beam`, `fiber` etc.). Use the key `newOptComp` and `addtoOptComp` to change the style for single components or for use with `\newpsobject` via the optional argument.



```
\begin{pspicture}(2.5,4)
  \psset{beam}
  % wrong, also beam width is changed
  \mirror[linewidth=3\pslinewidth](0,3)(2,3)(2,2)
  % correct result
  \addtopsstyle{OptComp}{linewidth=3\pslinewidth}
  \mirror(0,1)(2,1)(2,0)
\end{pspicture}
```

OptionalStyle ⟨*psstyle*⟩ default: `linestyle=dashed,dash=1.5pt 1pt`

This style is applied to components marked as `optional` in addition to `OptComp`.



```
\begin{pspicture}(3,1.5)
\lens[optional, beam](0,1)(3,1)
\end{pspicture}
```

`OptionalStyle` is applied after `OptComp` so that it can overwrite the general settings.

```
\begin{pspicture}(3,1.5)
\addtopsstyle{OptComp}{linewidth=3\pslinewidth, linecolor=red}
\newpsstyle{OptionalStyle}{linecolor=blue}
\lens(0,1)(2,1)
% linecolor overwritten, linewidth kept
\lens[optional](0,1)(3,1)
\end{pspicture}
```

VariableStyle ⟨*psstyle*⟩ default: `linewidth=0.8\pslinewidth, arrowinset=0, arrowscale=0.8, arrows=<->`

The style of the arrows of variable components (`\mirror` and `\optgrating`).

addtoOptComp=⟨*list*⟩

The OptComp style is extended by locally adding the parameters contained in ⟨*list*⟩. The ⟨*list*⟩ must be surrounded by curly braces.

newOptComp=⟨*list*⟩

Similar to addtoOptComp, but an existing OptComp style is overwritten with the new parameter set.

```
\begin{pspicture}(3,1.5)
  \psset{beam}
  \newpsstyle{OptComp}{linewidth=3\pslinewidth}
  \mirror[addtoOptComp={linestyle=dotted}](0.5,0)(0.5,1)(1.5,1)
  % overwrites the linewidth settings
  \mirror[newOptComp={linestyle=dotted}](1.5,1)(2.5,1)(2.5,0)
\end{pspicture}
```

optional=true, false default: false

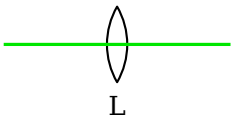
Mark an object as optional by applying the OptionalStyle style.

4. Free-ray components

This chapter describes all free-ray components and their options. Free-ray components differ from fiber components in that they provide transmittive or reflective interfaces which can be used to raytrace beams. They are divided in two types: some components like lenses need only two nodes for positioning (4.1–4.11), whereas e.g. mirrors require three nodes to determine the orientation (4.12–4.17).

4.1. Lens

`\lens`[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}

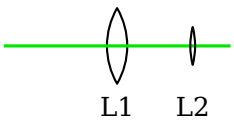



```
\begin{pspicture}(3,1.5)
  \lens[beam](0,1)(3,1){L}
\end{pspicture}
```

A lens is defined by its height and the radii of the input and output interfaces.

`lensheight`=⟨*num*⟩ default: 1

Set the height of the lens.



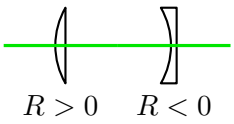

```
\begin{pspicture}(3,1.5)
  \lens[beam](0,1)(3,1){L1}
  \lens[lensheight=0.5](2,1)(3,1){L2}
\end{pspicture}
```

`lensradiusleft`=⟨*num*⟩ default: 1

Set the left radius of the lens. A positive ⟨*num*⟩ is for convex, a negative one for concave curvatures. Use zero for a plain surface.

`lensradiusright`=⟨*num*⟩ default: 1

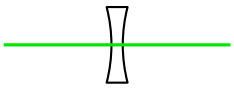
Same as `lensradiusleft` but for the right surface.




```
\begin{pspicture}(3,1.5)
  \psset{lensradiusright=0, beam}
  \lens[lensradiusleft=1](0,1)(1.5,1){$R > 0$}
  \lens[lensradiusleft=-1](1.5,1)(3,1){$R < 0$}
\end{pspicture}
```

`lensradius`=⟨*left*⟩[⟨*right*⟩]

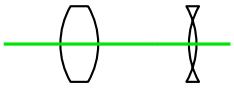
Set the left and right lens curvatures. If only a single value is given, it is used for both curvatures.



```
\begin{pspicture}(3,1.5)
  \lens[lensradius=-2, beam](0,1)(3,1)
\end{pspicture}
```

`lenswidth`=⟨*num*⟩ default: `0`

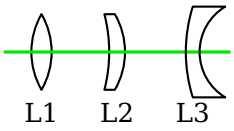
Usually only the height and the two radii are used to construct the lens, the width is calculated automatically. For `lenswidth` greater than zero, this width is used instead. This is only useful if you want to draw thick lenses and it can give ugly results if the value is choosen too small.




```
\begin{pspicture}(3,2)
  \lens[lenswidth=0.5, beam](0,1)(2,1){thicklens}
  \lens[lenswidth=0.1, beam](2,1)(3,1)
\end{pspicture}
```

`lens`=⟨*radiusleft*⟩[⟨*radiusright*⟩[⟨*height*⟩[⟨*width*⟩]]]

A convenience option to specify all lens parameters with a single option. Intermediate values can be skipped with a dot, values at the end can be omitted (e.g. `lens=. 1 1`).



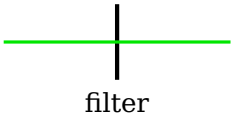

```
\begin{pspicture}(3,1.5)
  \lens[lens=1](0,1)(1,1){L1}
  \lens[lensradius=-2 1, beam](0,1)(3,1){L2}
  \lens[lens=2 -0.7 1.2](2,1)(3,1){L3}
\end{pspicture}
```

`thicklens`=`true`, `false` default: `false`

This option was necessary until version 2.1 in order to draw a thick lens of width `lenswidth` instead of using the calculated lens width. This option is superfluous since version 3.0 because setting `lenswidth` to a positive value automatically draws a thick lens.

4.2. Optical plate

\optplate[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,1.5)
  \optplate[beam](0,1)(3,1){filter}
\end{pspicture}
```

plateheight=⟨*num*⟩ default: 1

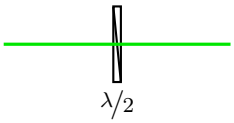
The height of the plate.

platelinewidth=⟨*num*⟩ or ⟨*dimen*⟩ default: 2\pslinewidth

The linewidth of the plate. This could be defined also with the generic linewidth option. But this parameter allows setting globally the linewidth of all plates.

4.3. Retardation plate

\optretplate[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,1.5)
  \optretplate[beam](0,1)(3,1){$\nicefrac{\lambda}{2}$}
\end{pspicture}
```

plateheight=⟨*num*⟩ default: 1

The height of the plate.

platewidth=⟨*num*⟩ default: 0.1

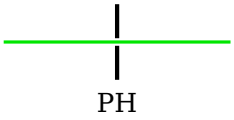
The width of the plate.

platesize=⟨*width*⟩ ⟨*height*⟩

The width and height of the plate, is equivalent to calling both platewidth and plateheight.

4.4. Pinhole

\pinhole[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,1.5)
  \pinhole[beam](0,1)(3,1){PH}
\end{pspicture}
```

outerheight=⟨*num*⟩ default: 1

The total height of the pinhole.

innerheight=⟨*num*⟩ default: 0.1

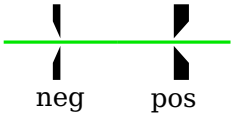
The height of the hole.

phlinewidth=⟨*num*⟩ or ⟨*dimen*⟩ default: 2\pslinewidth

The linewidth of the pinhole. This could be defined also with the generic linewidth option. But this parameter allows setting globally the linewidth of all pinholes.

phwidth=⟨*num*⟩ default: 0

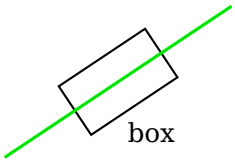
The pinhole is drawn in a more plastic style if phwidth is not zero, in this case the phlinewidth is ignored. For negative values the shape is mirrored.




```
\begin{pspicture}(3,1.5)
  \pinhole[beam, phwidth=-0.1](0,1)(1.5,1){neg}
  \pinhole[beam, phwidth=0.2](1.5,1)(3,1){pos}
\end{pspicture}
```

4.5. Box

\optbox[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,2)
  \optbox[beam](0,0)(3,2){box}
\end{pspicture}
```

optboxwidth=⟨*num*⟩ default: 1.4

The width of the box.

optboxheight=⟨*num*⟩ default: 0.8

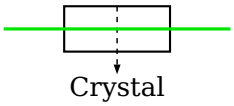
The height of the box.

optboxsize=⟨*width*⟩ ⟨*height*⟩

The width and height of the box, is equivalent to calling both optboxwidth and optboxheight.

4.6. Crystal

\crystal[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,1.3)
  \crystal[beam](0,1)(3,1){Crystal}
\end{pspicture}
```

crystalwidth=⟨*num*⟩ default: 1.4

The width of the crystal.

crystalheight=⟨*num*⟩ default: 0.6

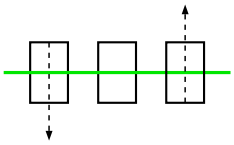
The height of the crystal.

crystalsize=⟨*width*⟩ ⟨*height*⟩

The width and height of the crystal, is equivalent to calling both crystalwidth and crystalheight.

caxislength=⟨*num*⟩ default: 0.3

The additional length of the c-axis, is dropped when set to 0 and is mirrored if the values is negative. The total length is caxislength plus crystalheight.



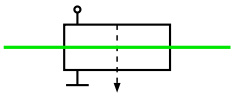
```
\begin{pspicture}(0,0.4)(3,1.5)
  \psset{crystalsize=0.5 0.8, label=-45 . l}
  \crystal[position=0.2, caxislength=0.5](0,1)(3,1){pos}
  \crystal[beam, caxislength=0](0,1)(3,1){0}
  \crystal[position=0.8, caxislength=-0.5](0,1)(3,1){neg}
\end{pspicture}
```

caxisinv=true, false default: false

Invert the direction of the c-axis, this is equivalent to changing the sign of caxislength.

voltage=true, false default: false

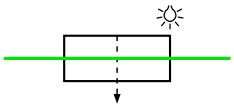
Draw a voltage connection and a ground symbol.



```
\begin{pspicture}(0,0.4)(3,1.5)
  \crystal[voltage, beam](0,1)(3,1)
\end{pspicture}
```

lamp=true, false default: false

Draw a lamp near the crystal.



```
\begin{pspicture}(0,0.4)(3,1.7)
  \crystal[lamp, beam](0,1)(3,1)
\end{pspicture}
```

lampscale=⟨*num*⟩

Scaling of the lamp. This parameter is deprecated since version 3.0, use style CrystalLamp instead.

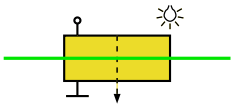
CrystalCaxis ⟨*psstyle*⟩ default: linewidth=0.7\pslinewidth, linestyle=dashed, dash=2pt 2pt, arrowinset=0, arrows=->

The style of the c-axis including the arrow type.

```
\begin{pspicture}(0,0.5)(3,1.6)
  \newpsstyle{CrystalCaxis}{linecolor=red, arrows=->>}
  \crystal[beam](0,1)(3,1)
\end{pspicture}
```

CrystalLamp ⟨*psstyle*⟩ default: linewidth=0.6\pslinewidth

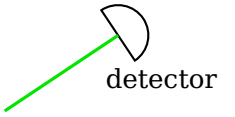
The style of the crystal background lamp.




```
\begin{pspicture}(3,1.7)
  \crystal[fillstyle=solid, fillcolor=yellow!90!black,
           label=1.2 -45, voltage,
           lamp, beam](0,1)(3,1){SBN:Ce}
\end{pspicture}
```

4.7. Detector

`\optdetector`[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,1.2)
  \optdetector[beam](0,0)(1.5,1){detector}
\end{pspicture}
```

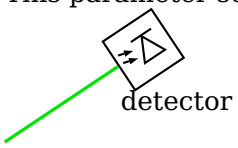
The detector is always placed at the end of the reference line (`position=end`). This setting cannot be changed.

`detsize`=⟨*num*⟩ or ⟨*width*⟩ ⟨*height*⟩ default: `0.8`

If a single number is given it is the side length (`diode`) or the diameter (`round`) of the detector. Two numbers define the width and height of the detector. Note, that `detsize=1` and `detsize=1 1` are not equivalent for `dettype=round`.

`dettype`=`round`, `diode` default: `round`

This parameter selects between different types of detectors.




```
\begin{pspicture}(3,1.5)
  \optdetector[beam, dettype=diode](0,0)(1.5,1){detector}
\end{pspicture}
```

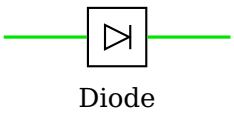
`DetectorStyle` ⟨*psstyle*⟩

The style of the diode for `dettype=diode`.

```
\begin{pspicture}(3,1.5)
  \newpsstyle{OptComp}{linewidth=2\pslinewidth}
  \newpsstyle{DetectorStyle}{linewidth=0.5\pslinewidth}
  \optdetector[dettype=diode](0,1)(2,1)
\end{pspicture}
```

4.8. Optical diode

`\optdiode`[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,2)
  \optdiode[beam](0,1)(3,1){Diode}
\end{pspicture}
```

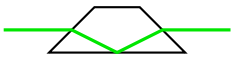
The optical diode has a default setting of `allowbeaminside=false`.

optdiodesize=⟨*num*⟩ default: 0.8

The side length of the optical diode.

4.9. Dove prism

\doveprism[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}

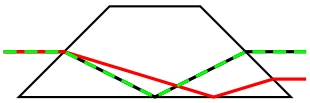


Dove

```
\begin{pspicture}(3,1.5)
  \doveprism[beam](0,1)(3,1){Dove}
\end{pspicture}
```

doveprismsize=⟨*num*⟩ or ⟨*width*⟩ ⟨*height*⟩ default: 0.6

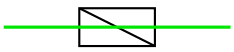
A single number defines the height of the prism, the total width is set to three times the height. Two number define the width and height of the dove prism. The angles at the input and output faces are always 45°.



```
\begin{pspicture}(4,1.5)
  \optplane(0,1)
  \doveprism[doveprismsize=1.2](0,1)(4,1)
  \optplane(4,1)
  \drawbeam[raytrace=false, linecolor=black]{-}
  \drawbeam[n=1.5, linecolor=red]{-}
  \drawbeam[n=*sqrt(5), linecolor=green, linestyle=dashed]{-}
\end{pspicture}
```

4.10. Glan-Thompson-Prisma

\glanthompson[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}



Glan-Thompson

```
\begin{pspicture}(3,1.5)
  \glanthompson[beam](0,1)(3,1){Glan-Thompson}
\end{pspicture}
```

glanthompsonwidth=⟨*num*⟩ default: 1.0

The width of the prism.

glanthompsonheight=⟨*num*⟩
default: 0.5

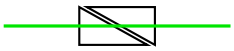
The height of the prism.

glanthompsonsize
=⟨*width*⟩ ⟨*height*⟩

The width and height of the prism, is equivalent to calling both glanthompsonwidth and glanthompsonheight.

glanthompsongap=⟨*num*⟩ default: 0

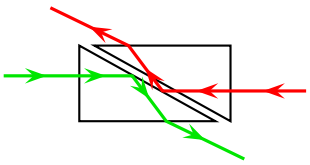
The separation of the two prisms in x-direction. The total width does not depend on this value, but the angle of the reflective plane is affected.



```
\begin{pspicture}(3,1)
  \glanthompson[glanthompsongap=0.1, beam](0,0.5)(3,0.5)
\end{pspicture}
```

The implementation of the Glan–Thompson prism for glanthompsongap=0 is equivalent to a beamsplitter which is positioned with two nodes only and which is not quadratic.

Please note, that the Prism cannot be used properly from both sides for glanthompsongap>0 because the current implementation allows only a single ambiguous interface (reflective or transmittive). In the following example you see, that the red beam is reflected at the wrong interface.



```
\begin{pspicture}(4,2)
  \pnode(0,1){A}\pnode(4,1){B}
  \glanthompson[glanthompsongap=0.2, glanthompsonsize=2
      1](A)(B)
  \optplane[angle=90](2,0)\optplane[angle=90](2,2)
  \addtopsstyle{Beam}{ArrowInside=->, arrowscale=1.5}
  \drawbeam[beampos=0.1](A){1-2}
  \drawbeam[beampos=0.1, linecolor=red](B){1}{3}
\end{pspicture}
```

Ex. 10 shows a further setup with a Glan–Thompson prism.

4.11. Polarization

\polarization[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}

```
\begin{pspicture}(3,1)
  \polarization[beam](0,0.5)(3,0.5)
\end{pspicture}
```

polsize=⟨*num*⟩ default: 0.6

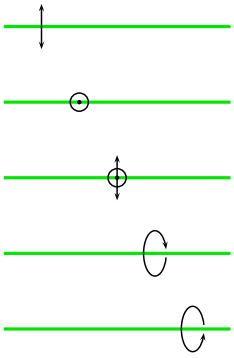
The size of the polarization symbol, the circle for perp and polmisc is half of this.

pollinewidth=⟨*num*⟩ or ⟨*dimen*⟩ default: 0.7\pslinewidth

The linewidth of the polarisation sign. This could be defined also with the generic linewidth option. But this parameter allows setting globally the linewidth of all polarisation signs. This parameter is deprecated since version 3.0, use style Polarization instead.

poltype=parallel, perp, misc, lcirc, rcirc default: parallel

This parameter choses the polarization type.



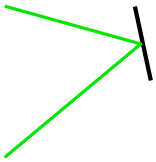
```
\begin{pspicture}(0,-0.3)(3,4.3)
  \psset[optexp]{beam}
  \begin{optexp}
  \polarization[poltype=parallel, abspos=0.5](0,4)(3,4)
  \polarization[poltype=perp, abspos=1](0,3)(3,3)
  \polarization[poltype=misc, abspos=1.5](0,2)(3,2)
  \polarization[poltype=lcirc, abspos=2](0,1)(3,1)
  \polarization[poltype=rcirc, abspos=2.5](0,0)(3,0)
  \end{optexp}
\end{pspicture}
```

Polarization ⟨*psstyle*⟩ default: arrowscale=0.8, linewidth=\pslinewidth, dotsize=3\pslinewidth

Affects the style of the polarisation symbols.

4.12. Mirror

\mirror[⟨*options*⟩](⟨*in*⟩)(⟨*center*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,2)
  \mirror[beam](0,0)(1.8,1.5)(0,2){M}
\end{pspicture}
```

mirrorwidth=⟨*num*⟩ default: 1

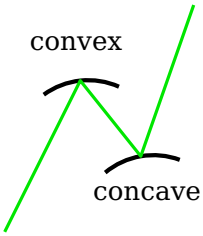
The width of the mirror.

mirrorlinewidth=⟨*num*⟩ or ⟨*dimen*⟩ default: 2\pslinewidth

The linewidth of the mirror. This could be defined also with the generic linewidth option. But this parameter allows setting globally the linewidth of all mirrors.

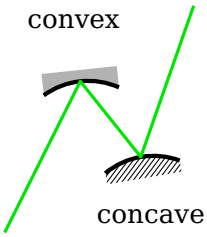
mirrorradius=⟨*radius*⟩[0] default: 0

This parameter defines the curvature of the mirror. A value of 0 is for a plain mirror, a negative radius for a convex mirror and a positive radius gives you a concave mirror.



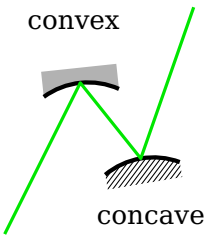

```
\begin{pspicture}(3,3)
  \psset[optexp]{labeloffset=0.5}
  \mirror[mirrorradius=1](0,0)(1,2)(1.8,1){convex}
  \mirror[mirrorradius=-1](1,2)(1.8,1)(2.5,3){concave}
  \drawbeam(0,0){-}(2.5,3)
\end{pspicture}
```

If no second value is specified, the curvature of the second interface of an extended or semitrans mirror is the same as that of the main interface (see previous example). If zero is given as second value (other values are not supported), the second interface is plain.




```
\begin{pspicture}(3,3)
  \mirror[mirrorradius=1 0, ↵
      mirrortype=semitrans](0,0)(1,2)(1.8,1){convex}
  \mirror[mirrorradius=-1 0, ↵
      mirrortype=extended](1,2)(1.8,1)(2.5,3){concave}
  \drawbeam(0,0){-}(2.5,3)
\end{pspicture}
```

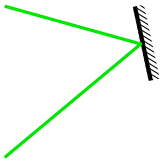
The value of mirrordepth always refers to the minimum depth of the mirror.




```
\begin{pspicture}(3,3)
  \psset[optexp]{mirrordepth=0.2}
  \mirror[mirrorradius=1 0,
      mirrortype=semitrans](0,0)(1,2)(1.8,1){convex}
  \mirror[mirrorradius=-1 0,
      mirrortype=extended](1,2)(1.8,1)(2.5,3){concave}
  \drawbeam(0,0){-}(2.5,3)
\end{pspicture}
```

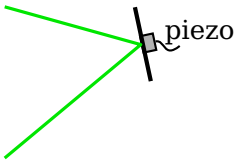
mirrortype=plain, piezo, extended, semitrans default: plain

The mirrortype selects between different types of mirrors. The style is controlled with PiezoMirror, ExtendedMirror, and SemitransMirror.



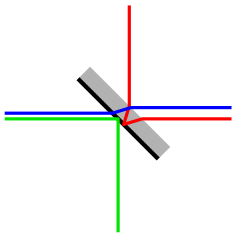

```
\begin{pspicture}(3,2)
  \mirror[mirrortype=extended,
          beam](0,0)(1.8,1.5)(0,2){ext}
\end{pspicture}
```

Note, that the default piece of wire is omitted when using extnode with a piezo mirror.

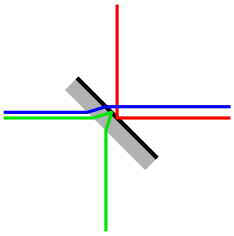



```
\begin{pspicture}(3,2)
\mirror[mirrortype=piezo,
        beam](0,0)(1.8,1.5)(0,2){piezo}
\end{pspicture}
```

A semi-transparent mirror (semitrans) has two interfaces, in contrast to the other mirror types. Their distance and position depends on the value and sign of mirrordepth.



```
\psset{unit=1.5}
\begin{pspicture}(2,2)
  \mirror[mirrortype=semitrans](0,1)(1,1)(1,0)
  \drawbeam(0,1){}(1,0)
  \drawbeam[linecolor=red](2,1){}(1,2)
  \drawbeam[beampos=0.05, linecolor=blue](0,1){}(2,1)
\end{pspicture}
```

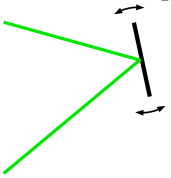


```
\psset{unit=1.5}
\begin{pspicture}(2,2)
  \mirror[mirrortype=semitrans, mirrordepth=-0.15](0,1)(1,1)(1,0)
  \drawbeam(0,1){}(1,0)
  \drawbeam[linecolor=red](2,1){}(1,2)
  \drawbeam[beampos=0.05, linecolor=blue](0,1){}(2,1)
\end{pspicture}
```

A negative mirrordepth is equivalent to a positive depth combined with angle=180.

variable=true, false default: false

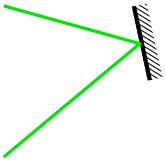
Draw an adjustable mirror which has two additional curved arrows on both sides. The appearance including the arrow types is determined by VariableStyle.




```
\begin{pspicture}(3,2)
  \mirror[beam, variable](0,0)(1.8,1.5)(0,2){var}
\end{pspicture}
```

mirrordepth=⟨*num*⟩ default: 0.1

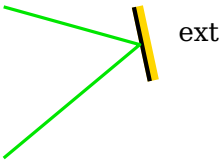
The total depth of an extended mirror.



```
\begin{pspicture}(3,2)
  \mirror[mirrortype=extended, mirrordepth=0.2,
          beam](0,0)(1.8,1.5)(0,2)
\end{pspicture}
```

ExtendedMirror ⟨*psstyle*⟩ default: linestyle=none, hatchwidth=0.5\pslinewidth, hatchsep=1.4\pslinewidth, fillstyle=hlines

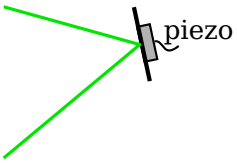
The style for the extended mirror.




```
\begin{pspicture}(3,2)
  \newpsstyle{ExtendedMirror}{fillstyle=solid,
                              fillcolor=Gold,
                              linestyle=none}
  \mirror[mirrortype=extended,
          beam](0,0)(1.8,1.5)(0,2){ext}
\end{pspicture}
```

PiezoMirror ⟨*psstyle*⟩ default: fillstyle=solid, fillcolor=black!30

This style defines the appearance of the piezo mirror. This can also be used to change the size of piezo part, as the example shows.



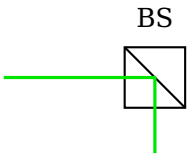

```
\begin{pspicture}(3,2)
  \addtopsstyle{PiezoMirror}{xunit=2}
  \mirror[mirrortype=piezo,
          beam](0,0)(1.8,1.5)(0,2){piezo}
\end{pspicture}
```

SemitransMirror ⟨*psstyle*⟩ default: `linestyle=none, fillstyle=solid, fillcolor=black!30`

This style defines the appearance of the `semitrans` mirror.

4.13. Beamsplitter

`\beamsplitter`[⟨*options*⟩](⟨*in*⟩)(⟨*center*⟩)(⟨*out*⟩){⟨*label*⟩}



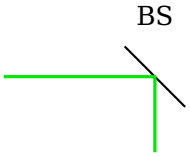

```
\begin{pspicture}(3,2)
  \beamsplitter[beam](0,1)(2,1)(2,0){BS}
\end{pspicture}
```

`bssize`=⟨*num*⟩ default: `0.8`

The beamsplitter size.

`bsstyle`=`cube`, `plate` default: `cube`

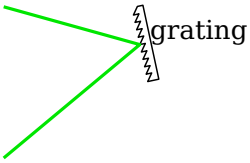
Selects between two types of beamsplitters: the beamsplitter cube (`cube`) and the semitransparent mirror (`plate`). This semitransparent mirror ist equivalent to `\mirror` with `mirrortype`=`semitrans` and `mirrordepth`=`0`.




```
\begin{pspicture}(3,2)
  \beamsplitter[bsstyle=plate, beam](0,1)(2,1)(2,0){BS}
\end{pspicture}
```

4.14. Optical grating

`\optgrating`[⟨*options*⟩](⟨*in*⟩)(⟨*center*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,2)
  \optgrating[beam](0,2)(1.8,1.5)(0,0){grating}
\end{pspicture}
```

Before package version 3.0 this component was called `\optgrid`. Since then this command name is deprecated and will be removed in future versions.

`gratingwidth`=⟨*num*⟩ default: `1`

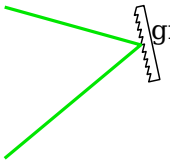
The width of the grating.

`gratingheight`=⟨*num*⟩ default: `0.15`

The total height of the grating.

`gratingdepth`=⟨*num*⟩ default: `0.075`

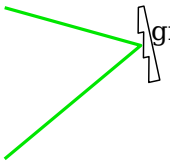
The modulation depth of the grating structure. The total height is adapted to this value if it is smaller than the modulation depth. Otherwise the total height is not affected.




```
\begin{pspicture}(3,2)
  \optgrating[gratingdepth=0.05,
              beam](0,2)(1.8,1.5)(0,0){grating}
\end{pspicture}
```

`gratingcount`=⟨*int*⟩ default: `10`

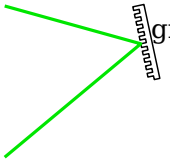
The number of grating grooves.




```
\begin{pspicture}(3,2)
  \optgrating[gratingcount=3,
              beam](0,2)(1.8,1.5)(0,0){grating}
\end{pspicture}
```

`gratingtype`=`blazed`, `binary` default: `blazed`

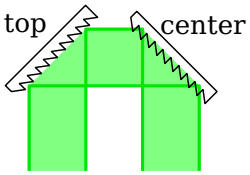
Select between a `binary` and a `blazed` grating.




```
\begin{pspicture}(3,2)
  \optgrating[gratingtype=binary,
              beam](0,2)(1.8,1.5)(0,0){grating}
\end{pspicture}
```

`gratingalign`=`t`, `top`, `c`, `center` default: `top`

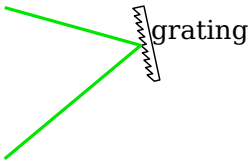
Selects if the reflection plane resides on `top` of the grating or in its `center`. If you choose `center`, you should use the `optexp` environment or a semitransparent beam filling.




```
\begin{pspicture}(3,2)
\pnode(0.5,0){A}\pnode(0.5,1.5){B}\pnode(2,1.5){C}\pnode(2,0){D}
\psset{unit=1.5, labeloffset=0.4}
\begin{optexp}
  \optgrating[gratingalign=top](A)(B)(C){top}
  \optgrating[gratingalign=center](B)(C)(D){center}
  \drawwidebeam[fillstyle=solid, fillcolor=green!50,
      beamwidth=0.5](A){-}(D)
\end{optexp}
\end{pspicture}
```

`reverse`=`true`, `false` default: `false`

Reverse the slope of the grooves of the `blazed` grating.




```
\begin{pspicture}(3,2)
  \optgrating[reverse, beam](0,2)(1.8,1.5)(0,0){grating}
\end{pspicture}
```

`gratinglinewidth`=⟨*num*⟩ or ⟨*dimen*⟩ default: `0.7\pslinewidth`

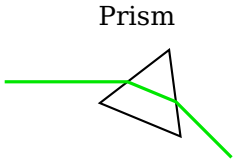
The linewidth of the grating. This could be defined also with the generic `linewidth` option. But this parameter allows setting globally the linewidth of all gratings.

`optgridwidth`
`optgridheight`
`optgriddepth`
`optgridcount`
`optgridtype`
`optgridlinewidth`

These parameters were renamed to the respective `grating`∗ parameters and are deprecated since version 3.0.

4.15. Prism

`\optprism`[⟨*options*⟩](⟨*in*⟩)(⟨*center*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,2)
  \optprism[beam](0,1)(2,1)(3,0){Prism}
\end{pspicture}
```

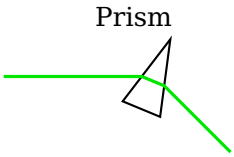
The prism is always placed symmetric between ⟨*in*⟩ and ⟨*out*⟩ nodes. For asymmetric beam traces see Sec. 8.

prismsize=⟨num⟩ default: 1

The height of the prism.

prismangle=⟨num⟩ default: 60

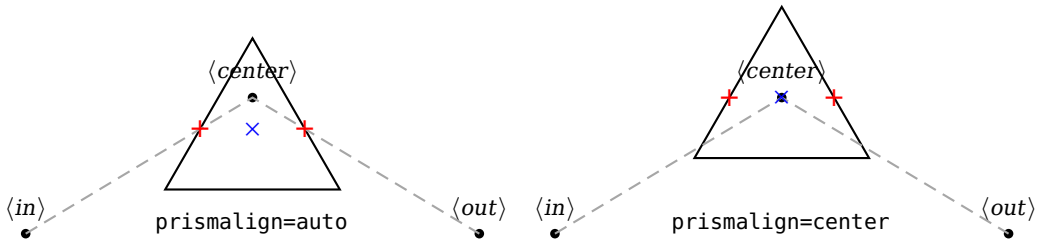
The upper angle of the prism.




```
\begin{pspicture}(3,2)
  \optprism[prismangle=30, beam](0,1)(2,1)(3,0){Prism}
\end{pspicture}
```

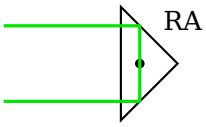
prismalign=auto, center default: auto

Sets the vertical alignment of the prism with respect to the “reflection” node ⟨center⟩. For auto, the prism is shifted such, that the interface nodes lay on the connection between the respective reference node and the “reflection” node. If the value is center, the “reflection” node coincides with the component center. See the following examples for further explanation.



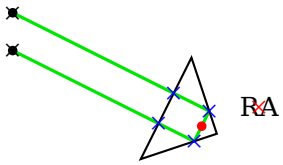
4.16. Right-angle prism

\rightangleprism[⟨options⟩](⟨in⟩)(⟨center⟩)(⟨out⟩){⟨label⟩}




```
\begin{pspicture}(3,2)
  \psdot(1.8,1)
  \rightangleprism[beam](0,1.5)(1.8,1)(0,0.5){RA}
\end{pspicture}
```

The right-angle prisms is align such that the incoming and reflected beam are parallel and the ⟨center⟩ node is vertically centered in the prism.



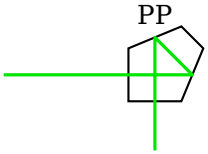

```
\begin{pspicture}(3.5,2)
  \pnode(0,2){A}\pnode(2.5,0.5){G}\pnode(0,1.5){B}
  \begin{optexp}
  \rightangleprism[beam, showifcnodes, showoptdots](A)(G)(B){RA}
  \end{optexp}
\end{pspicture}
```

raprismsize=⟨*num*⟩ default: 1.5

The length of the input plane.

4.17. Penta prism

\pentaprism[⟨*options*⟩](⟨*in*⟩)(⟨*center*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,2)
  \pentaprism[beam](0,1)(2,1)(2,0){PP}
\end{pspicture}
```

pentaprismsize=⟨*num*⟩ default: 0.7

The length of the input and output plane.

5. Fiber components

This chapter describes all fiber components and their options. They differ from free-ray components in that they are connected by default with fibers to their reference nodes, and that they cannot be used for raytracing. Instead of automatic fiber connections the components can also be connected manually with fibers, see Sec. 8.6.

Most components require only two reference nodes and are handled like the free-ray dipoles. Some special components like couplers (Sec. 5.10) and circulators (Sec. 5.9) are treated differently.

`usefiberstyle`=`true`, `false`

For some components (e.g. `\optfilter` or `\optmzm`) it can be nice to highlight some internals. For example, if this option is enabled the passing parts of the optical filter are drawn with the `Fiber` style. In the documentation this parameter is enabled to show the parts which are affected.

5.1. Optical fiber

`\optfiber`[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}


```
\begin{pspicture}(3,1.5)
  \optfiber[label=0.3](0,0.5)(3,0.5){SSMF}
\end{pspicture}
```

`fiberloops`=⟨*int*⟩ default: 3

Number of the fiber loops.


```
\begin{pspicture}(3,1.5)
  \optfiber[fiberloops=2, label=0.3](0,0.5)(3,0.5){SSMF}
\end{pspicture}
```

fiberloopradius=⟨*num*⟩ default: 0.4

Radius of the fiber loops.


```
\begin{pspicture}(3,1.5)
  \optfiber[fiberloopradius=0.2, label=0.3](0,0.5)(3,0.5){SSMF}
\end{pspicture}
```

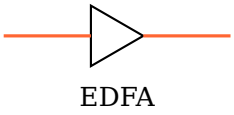
fiberloopsep=⟨*num*⟩ default: 0.3

Separation between two successive fiber loops.


```
\begin{pspicture}(3,1.5)
  \optfiber[fiberloopsep=0.6, label=0.3](0,0.5)(3,0.5){SSMF}
\end{pspicture}
```

5.2. Optical amplifier

\optamp[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}



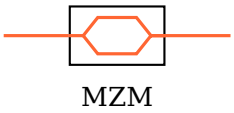

```
\begin{pspicture}(3,1.5)
  \optamp(0,1)(3,1){EDFA}
\end{pspicture}
```

optampsize=⟨*num*⟩ or ⟨*width*⟩ ⟨*height*⟩ default: 0.8

A single number gives the side length of the amplifier, two numbers the width and height. Note, that `optampsize=1` and `optampsize=1 1` do not give the same result. You may also change the relation of width and height with `xunit` and `yunit`.

5.3. Mach–Zehnder modulator

\optmzm[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}



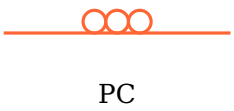

```
\begin{pspicture}(3,1.5)
  \optmzm(0,1)(3,1){MZM}
\end{pspicture}
```

optmzmsize=⟨*num*⟩ or ⟨*width*⟩ ⟨*height*⟩ default: 0.8

A single number gives the modulator height, the width is `1.6` times the height. Two numbers define width and height directly.

5.4. Polarization controller

`\polcontrol`[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}



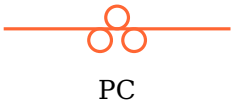

```
\begin{pspicture}(3,1.5)
  \polcontrol(0,1)(3,1){PC}
\end{pspicture}
```

`polcontrolsize`=⟨*num*⟩ default: `0.15`

The radius of the polarization controller circles.

`polcontroltype`=`linear`, `triangle` default: `linear`

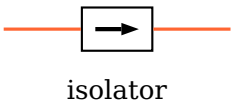
The type of polarization controller.




```
\begin{pspicture}(3,1.5)
  \polcontrol[polcontroltype=triangle](0,1)(3,1){PC}
\end{pspicture}
```

5.5. Isolator

`\optisolator`[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}



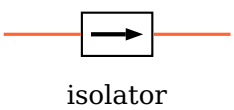

```
\begin{pspicture}(3,1.5)
  \optisolator(0,1)(3,1){isolator}
\end{pspicture}
```

`isolatorsize`=⟨*num*⟩ or ⟨*width*⟩ ⟨*height*⟩ default: `0.6`

A single number gives the isolator height, the width is `1.6` times the height. Two numbers define width and height directly.

`IsolatorArrow` ⟨*psstyle*⟩ default: `linewidth=2\pslinewidth, arrowinset=0`

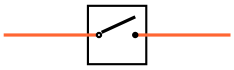
The style of the isolator arrow. This can be especially useful to adapt the length of the arrow.




```
\begin{pspicture}(3,1.5)
  \addtopsstyle{IsolatorArrow}{xunit=1.2}
  \optisolator(0,1)(3,1){isolator}
\end{pspicture}
```

5.6. Optical switch

\optswitch[⟨options⟩](⟨in⟩)(⟨out⟩){⟨label⟩}



opened switch

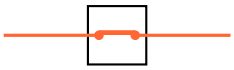
```
\begin{pspicture}(3,1.5)
  \optswitch(0,1)(3,1){opened switch}
\end{pspicture}
```

switchsize=⟨num⟩ or ⟨width⟩ ⟨height⟩ default: 0.8

A single number defines the side length of the switch, otherwise the height and width can be specified separately.

switchstyle=opened, closed default: opened

Indicate the switch as opened or closed.

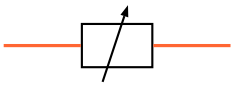


closed switch

```
\begin{pspicture}(3,1.5)
  \optswitch[switchstyle=closed](0,1)(3,1){closed switch}
\end{pspicture}
```

5.7. Fiber delay line

\fiberdelayline[⟨options⟩](⟨in⟩)(⟨out⟩){⟨label⟩}



delay line

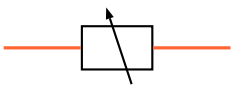
```
\begin{pspicture}(3,1.5)
  \fiberdelayline(0,1)(3,1){delay line}
\end{pspicture}
```

fdlsize=⟨num⟩ or ⟨width⟩ ⟨height⟩ default: 0.6

A single number gives the component height, the width is 1.6 times the height. Two numbers define width and height directly.

FdlArrow ⟨psstyle⟩ default: arrowinset=0, arrows=->

The style of the arrow. This can be especially useful to adapt the length of the arrow which may not be appropriate depending on the fdlsize, or to mirror the arrow.

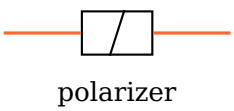


delay line

```
\begin{pspicture}(3,1.5)
  \addtopsstyle{FdlArrow}{xunit=-1}
  \fiberdelayline(0,1)(3,1){delay line}
\end{pspicture}
```

5.8. Polarizer

\optfiberpolarizer[⟨options⟩](⟨in⟩)(⟨out⟩){⟨label⟩}



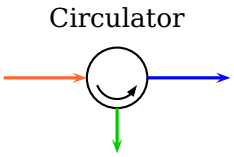

```
\begin{pspicture}(3,1.5)
  \optfiberpolarizer(0,1)(3,1){polarizer}
\end{pspicture}
```

fiberpolsize=⟨num⟩ or ⟨width⟩ ⟨height⟩ default: 0.6

A single number gives the component height, the width is 1.6 times the height. Two numbers define width and height directly.

5.9. Optical circulator

\optcirculator(⟨left⟩)(⟨right⟩)(⟨bottom⟩){⟨label⟩}




```
\begin{pspicture}(3,2)
  \addtopsstyle{Fiber}{arrows=->}
  \addtopsstyle{FiberOut1}{linecolor=blue}
  \addtopsstyle{FiberOut2}{linecolor=green!80!black}
  \optcirculator(0,1)(3,1)(1.5,0){Circulator}
\end{pspicture}
```

optcircsize=⟨num⟩ default: 0.8

The diameter of the circulator.

optcircangleA=⟨num⟩ default: -160

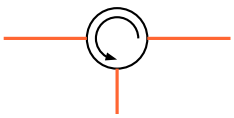
The starting angle of the internal arrow.

optcircangleB=⟨num⟩ default: -20

The ending angle of the internal arrow.

optcircangle=⟨num⟩ ⟨num⟩

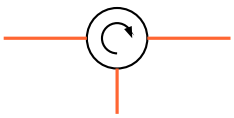
Short notation to specifying both angles.



```
\begin{pspicture}(3,1.5)
  \optcirculator[optcircangle=0 -90](0,1)(3,1)(1.5,0)
\end{pspicture}
```

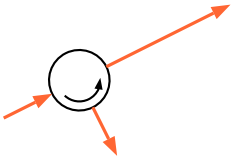
OptCircArrow ⟨*psstyle*⟩ default: `unit=0.7, arrows=->, arrowinset=0`

The style of the internal arrow. It specifies the direction of the arrow and the size of the arc.



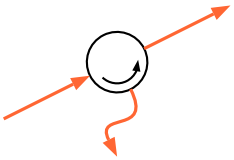
```
\begin{pspicture}(3,1.5)
  \newpsstyle{OptCircArrow}{unit=0.5, arrows=<-, arrowinset=0}
  \optcirculator[optcircangle=0 -90](0,1)(3,1)(1.5,0)
\end{pspicture}
```

The circulator is positioned by default such that the input and output fibers are orthogonal to each other, like shown in the example:



```
\begin{pspicture}(3,2)
  \addtopsstyle{Fiber}{arrowscale=1.3, arrows=->, arrowinset=0}
  \optcirculator(0,0.5)(3,2)(1.5,0)
\end{pspicture}
```

The positioning parameters (Sec. 3.2) refer to the reference nodes ⟨*left*⟩ and ⟨*right*⟩ as usual and are not related to the automatic position. This implies, that the default position is not equivalent to `position=0.5`.

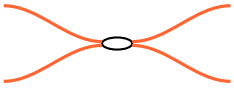


```
\begin{pspicture}(3,2)
  \addtopsstyle{Fiber}{arrowscale=1.3, arrows=->, arrowinset=0,
      ncurv=1.5}
  \optcirculator[position=0.5](0,0.5)(3,2)(1.5,0)
\end{pspicture}
```

5.10. Fiber coupler

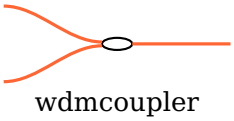
The package provides three fiber couplers which all share the same shapes and parameters.

`\optcoupler`(⟨*tl*⟩)(⟨*bl*⟩)(⟨*tr*⟩)(⟨*br*⟩){⟨*label*⟩}



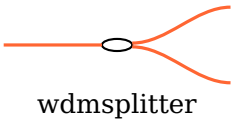

```
\begin{pspicture}(0,-0.4)(3,1)
  \optcoupler(0,1)(0,0)(3,1)(3,0){optcoupler}
\end{pspicture}
```

`\wdmcoupler`(⟨*tl*⟩)(⟨*bl*⟩)(⟨*r*⟩){⟨*label*⟩}




```
\begin{pspicture}(0,-0.4)(3,1)
  \wdmcoupler(0,1)(0,0)(3,0.5){wdmcoupler}
\end{pspicture}
```

`\wdmsplitter`(⟨*l*⟩)(⟨*tr*⟩)(⟨*br*⟩){⟨*label*⟩}



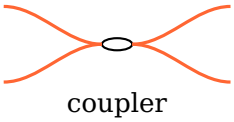

```
\begin{pspicture}(0,-0.4)(3,1)
  \wdmsplitter(0,0.5)(3,1)(3,0){wdmsplitter}
\end{pspicture}
```

`couplersize`=⟨*num*⟩ or ⟨*width*⟩ ⟨*height*⟩ default: `0.2`

For a single number the width is twice this value, the height `0.8` times this value. Two numbers define width and height directly.

`couplersep`=⟨*num*⟩ default: `0.1`

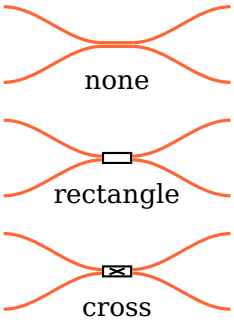
The vertical distance between two fiber ports.




```
\begin{pspicture}(0,-0.5)(3,1)
  \optcoupler[couplersep=0](0,1)(0,0)(3,1)(3,0){coupler}
\end{pspicture}
```

`couplertype`=`none`, `ellipse`, `rectangle`, `cross` default: `ellipse`

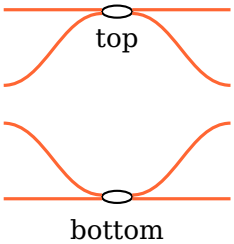
Select between different coupler types.




```
\begin{pspicture}(3,4.5)
  \psset{labeloffset=0.5}
  \optcoupler[couplertype=none]%
      (0,4.5)(0,3.5)(3,4.5)(3,3.5){none}
  \optcoupler[couplertype=rectangle]%
      (0,3)(0,2)(3,3)(3,2){rectangle}
  \optcoupler[couplertype=cross]%
      (0,1.5)(0,0.5)(3,1.5)(3,0.5){cross}
\end{pspicture}
```

`coupleralign`=`t`, `top`, `b`, `bottom`, `c`, `center` default: `center`

The alignment of the coupler with respect to the reference nodes. Note, that the exact position of the coupler center and accordingly that of the label depends on this setting (see 7.3).

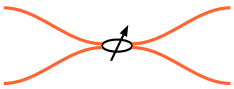



```
\begin{pspicture}(3,3)
  \psset{labeloffset=0.4}
  \optcoupler[coupleralign=top]%
            (0,3)(0,2)(3,3)(3,2){top}
  \optcoupler[coupleralign=bottom]%
            (0,1.5)(0,0.5)(3,1.5)(3,0.5){bottom}
\end{pspicture}
```

align=top, bottom, center default: center

This parameter was renamed in version 3.0 to coupleralign and is deprecated.

variable=true, false default: true



```
\begin{pspicture}(3,1)
  \optcoupler[variable](0,1)(0,0)(3,1)(3,0)
\end{pspicture}
```

VariableCoupler

⟨*psstyle*⟩ default: arrowinset=0, arrows=->

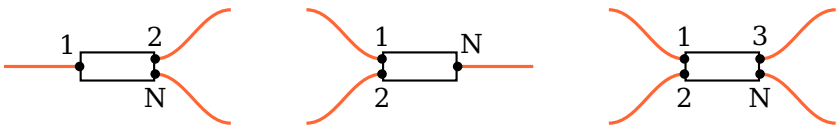
The style of the arrow of the variable coupler.

Every coupler has a variant which needs only two nodes, one input and one output node. For \optcoupler, both nodes are used twice, \wdmsplitter uses node ⟨*out*⟩ twice, and \wdmcoupler uses node ⟨*in*⟩ twice. This can be very useful if you do not use the automatic connections but connect the components manually.

```
\begin{pspicture}(4,2)
  \pnode(0,1){A}\pnode(4,1){B}
  \psset[optexp]{fiber=none}
  \wdmsplitter[position=0.2, coupleralign=b](A)(B)
  \wdmcoupler[position=0.8, coupleralign=t](A)(B)
  \drawfiber{(A)}{1}{2}{(B)}
  \drawfiber{1}{([offset=0.5, Xnodesep=1]\oenodeIfc{2}{1})}
  \drawfiber{([offset=-0.5, Xnodesep=-1]\oenodeIfc{2}{2})}{2}
\end{pspicture}
```

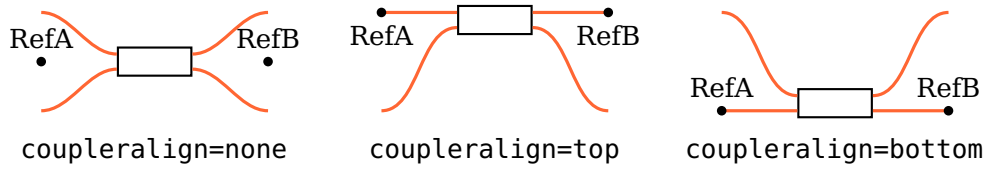
5.10.1. Input and output nodes

The definition of input and output nodes from Sec. 7.6 cannot be applied to couplers. Here, the nodes are simply numbered from 1 (left top) to N (right bottom).



5.10.2. Reference nodes

The definition of reference nodes from Sec. 7.2 cannot be applied to couplers and depends on `coupleralign`.



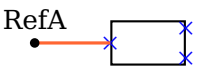
5.11. Fiber box

`\fiberbox`(⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}

`\fiberbox`(⟨*tl*⟩)(⟨*bl*⟩)(⟨*tr*⟩)(⟨*br*⟩){⟨*label*⟩}

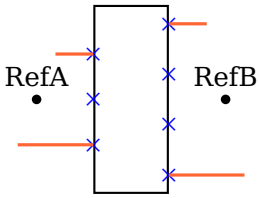
A `\fiberbox` can be positioned either with two, or four nodes which affects the positioning of the component and the interface nodes, and the automatic fiber connections.

If only two nodes are used these are the reference nodes and the separation between the components input and output nodes must be specified manually (see `fiberboxsepin` and `fiberboxsepout`). Only the side of the component is connected automatically which has a single node only (see `fiberboxcount` and the following example).




```
\begin{pspicture}(3,2)
  \fiberbox[fiberboxcount=1x2, showifcnodes](0,1)(3,1)
\end{pspicture}
```

When four nodes are used the interface nodes and their separation are determined automatically. The component is positioned between the center of the two input and the center of the two output nodes. The first input interface node is aligned with the ⟨*tl*⟩ node, the last input node is aligned with ⟨*bl*⟩, all other input nodes are equally distributed between these. The output nodes are handled accordingly. The four specified nodes are connected automatically to the component, each with the interface node it is aligned to.



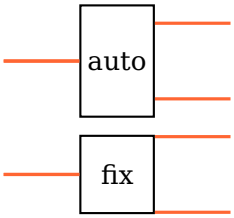

```
\begin{pspicture}(3,2)
  \fiberbox[fiberboxcount=3x4, ↵
      showifcnodes](0.5,1.6)(0,0.4)(2.5,2)(3,0)
\end{pspicture}
```

fiberboxwidth=⟨*num*⟩ default: 1

The width of the fiber box.

fiberboxheight=⟨*num*⟩ default: 0

The height of the fiber box. If the height is lower than the space required by the input or output nodes, it is calculated automatically. The calculated height is $\max(\text{sepin} \times (\text{N}-1), \text{sepout} \times (\text{M}-1))$.




```
\begin{pspicture}(3,2.8)
  \psset[optexp]{innerlabel, fiberboxwidth=1, fiberboxcount=1x2}
  \fiberbox(0,2)(0,2)(3,2.5)(3,1.5){auto}
  \fiberbox[fiberboxheight=1.05](0,0.5)(0,0.5)(3,1)(3,0){fix}
\end{pspicture}
```

fiberboxsize=⟨*width*⟩ ⟨*height*⟩

The width and height of the fiber box, is equivalent to calling both fiberboxwidth and fiberboxheight.

fiberboxsepin=⟨*num*⟩ default: 0.2

The separation between two input nodes. This value is used only if the fiber box is positioned with two nodes. Otherwise the separation is calculated automatically.

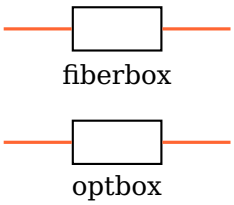
fiberboxsepout=⟨*num*⟩ default: 0.2

Equivalent to fiberboxsepout, but for the output nodes.

fiberboxcount=⟨*N*⟩x⟨*M*⟩ default: 2x2

Number of input (⟨*N*⟩) and output nodes (⟨*M*⟩).

For certain parameter configurations a \fiberbox is equivalent to an \optbox.




```
\begin{pspicture}(0,0.3)(3,2.6)
  \psset[optexp]{label=0.6}
  \fiberbox[fiberboxsize=1.2 0.6,
      fiberboxcount=1x1](0,2.5)(3,2.5){fiberbox}
  \optbox[optboxsize=1.2 0.6, fiber](0,1)(3,1){optbox}
\end{pspicture}
```

6. Hybrid components

This chapter describes the components which can be used both for free-ray and fiber optics but which differ from these two categories.

The optical filter (\optfilter) is connected by default with automatic fiber connections, but can likewise be used with free-ray beams, in contrast to the fiber components (Sec. 5). The fiber collimator (\fibercollimator) has one interface only, the other connection is a fiber.

6.1. Optical filter

\optfilter[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}

bandpass

```
\begin{pspicture}(3,1.5)
  \optfilter(0,1)(3,1){bandpass}
\end{pspicture}
```

filtersize=⟨*num*⟩ default: 0.8

The size of the filter.

filtertype=bandpass, bandstop, lowpass, highpass default: bandpass

Select between different filter types.

bandstop

```
\begin{pspicture}(3,1.5)
  \optfilter[filtertype=bandstop](0,1)(3,1){bandstop}
\end{pspicture}
```

lowpass

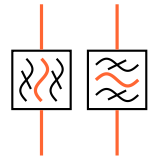
```
\begin{pspicture}(3,1.5)
  \optfilter[filtertype=lowpass](0,1)(3,1){lowpass}
\end{pspicture}
```

highpass

```
\begin{pspicture}(3,1.5)
  \optfilter[filtertype=highpass](0,1)(3,1){highpass}
\end{pspicture}
```

`filterangle`=⟨*num*⟩ default: `0`

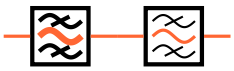
Rotates the “inner” part of the filter relativ to its frame. Alternatively `innercompalign` can be used.



```
\begin{pspicture}(2,2)
  \optfilter(0.5,0)(0.5,2)
  \optfilter[filterangle=90](1.5,0)(1.5,2)
\end{pspicture}
```

`FilterStyle` ⟨*psstyle*⟩

Change the style of the internal filter lines.



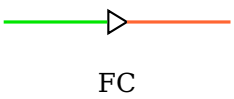
```
\begin{pspicture}(3,1.5)
  \newpsstyle{OptComp}{linewidth=2\pslinewidth}
  \optfilter(0,1)(1.5,1)
  \newpsstyle{FilterStyle}{linewidth=0.5\pslinewidth}
  \optfilter(1.5,1)(3,1)
\end{pspicture}
```

Usage as free-ray component, `allowbeaminside` is set to `false` by default:

```
\begin{pspicture}(3,1.5)
  \optfilter[fiber=none](0,1)(3,1)
  \addtopsstyle{Beam}{beamwidth=0.3,
     fillstyle=solid, fillcolor=green, opacity=0.2}
  \drawwidebeam(0,1){}(3,1)
\end{pspicture}
```

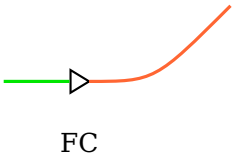
6.2. Fiber collimator

`\fibercollimator`(⟨*in*⟩)(⟨*A*⟩)(⟨*B*⟩)(⟨*out*⟩){⟨*label*⟩}



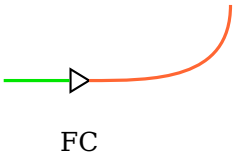

```
\begin{pspicture}(3,1.5)
  \fibercollimator[beam](0,1)(3,1){FC}
\end{pspicture}
```

The fiber collimator can be used with two, three or four points. With two points, the collimator is placed like any other dipole component between ⟨*in*⟩ and ⟨*out*⟩ node. For three nodes, the fiber is drawn as `\psbezier` curve for which the central node ⟨*A*⟩ is used twice. Positioning parameters (see Sec. 3.2) can be used to shift the object between ⟨*in*⟩ and ⟨*A*⟩ nodes.




```
\begin{pspicture}(3,2)
  \fibercollimator[beam](0,1)(2,1)(3,2){FC}
\end{pspicture}
```

For four nodes, the fiber is drawn as `\psbezier` curve with the specified nodes. Positioning parameters (siehe Sec. 3.2) can be used to shift the object between the first two nodes (⟨*in*⟩ and ⟨*A*⟩).




```
\begin{pspicture}(3,2)
  \fibercollimator[beam](0,1)(2,1)(3,1)(3,2){FC}
\end{pspicture}
```

`fibercolsize`=⟨*num*⟩ or ⟨*width*⟩ ⟨*height*⟩ default: `0.3`

A single number gives the side length of the collimator, two numbers the width and height. Note, that `fibercolsize=1` and `fibercolsize=1 1` do not give the same result. You may also change the relation of width and height with `xunit` and `yunit`.

7. Special nodes

Every `pst-optexp` object of an experimental setup provides several special nodes which are related to its geometry and positioning. They can be accessed and used for related positioning and drawing.

You should always use the dedicated macros to access these node names.

`\oenode`{⟨*node*⟩}{⟨*comp*⟩}

This is the basic command to access any node associated with a certain component. The first argument ⟨*node*⟩ is the identifier of the requested node. The second argument ⟨*comp*⟩ is the name of the target component (according to Sec. 7.1). If left empty, it uses the last component defined.

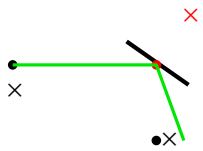
For most special nodes an appropriate macro is provided, which you are strongly advised to use, because the naming conventions may change. Using the macros makes sure, that you always get the correct node names. Therefore, the available identifiers are not listed explicitely.

`namingscheme`=old, new default: `new`

This option is for backward compatibility only. In version 2.1 special component nodes had to be accessed by their explicit name. You should use this option only if you accessed internal nodes directly in old documents. Since version 3.0 explicit macros are provided to access all special component nodes, so that the actual naming scheme does not matter.

`showoptdots`=`true`, `false` default: `false`

Draw some special component nodes for debugging: The black points are the normal and the black crosses are the transformed reference nodes (Sec. 7.2), the red point is the center node (Sec. 7.3), and the red cross is the label node (Sec. 7.4).



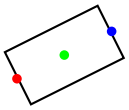
```
\begin{pspicture}(3,2)
\mirror[showoptdots, angle=10, beam](0,1)(1.9,1)(1.9,0)
\end{pspicture}
```

7.1. Component identifiers

All components of a setup drawing are numbered automatically in increasing order according to their definition in the code, starting with 1. The components as well as their special nodes can always be accessed by this number (ID).

compname=⟨*string*⟩

Assigns a name identifier to a component, which can then be referenced both by this name and by its ID. The parameter can be assigned only in the optional Argument of a `pst-optexp` component and should be unique within one `pspicture` environment.



```
\begin{pspicture}(2,2)
  \optbox[compname=MyBox](0,1)(2,2)
  \psdot[linecolor=red](\oenodeIn{MyBox})% use the compname
  \psdot[linecolor=blue](\oenodeOut{1}) % use the ID
  \psdot[linecolor=green](\oenodeCenter{}) % the last component
\end{pspicture}
```

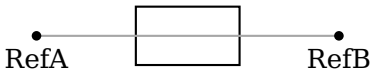
7.2. Reference nodes

\oenodeRefA{⟨*comp*⟩}

\oenodeRefB{⟨*comp*⟩}

The input and output reference nodes.

These are the original nodes which were used for the component positioning, `\oenodeRefA` is the first and `\oenodeRefB` the last node. This definition is not valid couplers (5.10.2) and circulators (5.9).



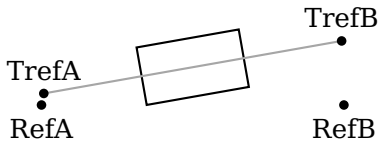

```
\begin{pspicture}(5,1)
  \pnode(0.5,0.5){A}\pnode(4.5,0.5){B}
  \optbox(A)(B)
  \psline[style=Refline](\oenodeRefA{})(\oenodeRefB{})
  \psdot(\oenodeRefA{})\uput[-90](\oenodeRefA{}){RefA}
  \psdot(\oenodeRefB{})\uput[-90](\oenodeRefB{}){RefB}
\end{pspicture}
```

\oenodeTrefA{⟨comp⟩}

\oenodeTrefB{⟨comp⟩}

The transformed input and output reference nodes.

These are the input and output reference nodes which are transformed together with the component according to the compshift and angle parameters.

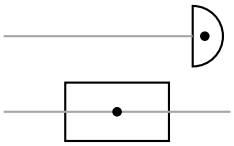



```
\begin{pspicture}(5,1.8)
  \pnode(0.5,0.5){A}\pnode(4.5,0.5){B}
  \optbox[compshift=0.5, angle=10](A)(B)
  \psdot(\oenodeRefA{})\uput[-90](\oenodeRefA{}){RefA}
  \psdot(\oenodeRefB{})\uput[-90](\oenodeRefB{}){RefB}
  \psdot(\oenodeTrefA{})\uput[90](\oenodeTrefA{}){TrefA}
  \psdot(\oenodeTrefB{})\uput[90](\oenodeTrefB{}){TrefB}
\end{pspicture}
```

7.3. Center node

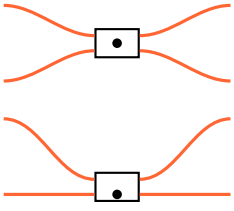
\oenodeCenter{⟨comp⟩}

This node lays, except for a few components like e.g. \optcirculator and \optcoupler (see below), in the center of the component.



```
\begin{pspicture}(3,2)
  \optbox(0,0.5)(3,0.5)
  \optdetector(0,1.5)(2.5,1.5)
\end{pspicture}
```

For \optcoupler the position of the component center depends on the coupleralign settings.

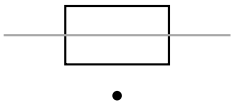


```
\begin{pspicture}(0,0.3)(3,3)
  \psset[optexp]{couplersize=0.6 0.4, couplersep=0.2,
     couplertype=rectangle}
  \optcoupler(0,3)(0,2)(3,3)(3,2)
  \psdot(\oenodeCenter{})
  \optcoupler[coupleralign=b](0,1.5)(0,0.5)(3,1.5)(3,0.5)
  \psdot(\oenodeCenter{})
\end{pspicture}
```

7.4. Label node

\oenodeLabel{⟨*comp*⟩}

The component label is placed at this node. It is also available if no label was specified. The label node is identical with the center node for labeloffset=0.



```
\begin{pspicture}(3,1.5)
  \optbox(0,1)(3,1)
\end{pspicture}
```

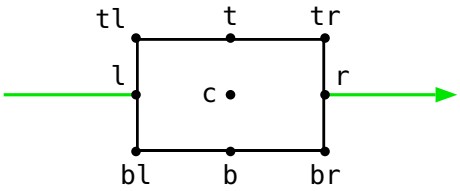
7.5. External nodes

\oenodeExt{⟨*comp*⟩}

An external node can be placed at different positions along the component boundary. It gets defined only for external usage and does not affect the component in any way.

extnode=⟨*refpoint*⟩

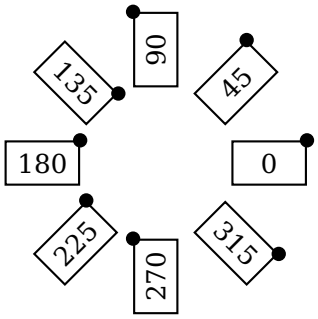
Set the position of the external node. Like the reference point of \rput, can be any comination of c (center), t (top), b (bottom), l (left), and r (right). Not all components support any possible combination. The allowed positions of each component are listed in Sec. 11.2.1.



extnodealign=rel, relative, abs, absolute default: abs

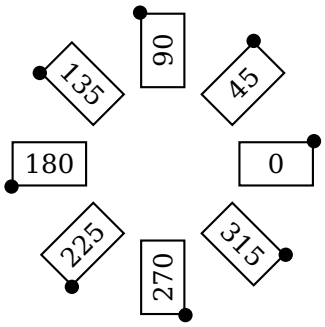
The definition of “top” (t) and the other extnode refpoint parameters can be absolute or relative to the component.

In the following example the external node is always placed “top right” independent of the actual order of the reference nodes (extnodealign=abs). The behaviour is identical to the label rotation for labelref=relative.




```
\begin{pspicture}(-2,-2)(2,2)
  \psset{endbox, optboxsize=1 0.6, dotscale=1.5}
  \psset{extnodealign=abs, extnode=tr}
  \multido{\i=0+45}{8}{%
    \optbox[innerlabel](0,0)(1;\i){\i}
    \psdot(\oenodeExt{})
  }
\end{pspicture}
```

In the next example the position is relative to the order of input and output reference node (`extnodealign=rel`).




```
\begin{pspicture}(-2,-2)(2,2)
  \psset{endbox, optboxsize=1 0.6, dotscale=1.5}
  \psset{extnodealign=rel, extnode=tr}
  \multido{\i=0+45}{8}{%
    \optbox[innerlabel](0,0)(1;\i){\i}
    \psdot(\oenodeExt{})
  }%
\end{pspicture}
```

7.6. Interface nodes

`\oenodeIfc{`⟨*num*⟩`}{`⟨*comp*⟩`}`

The interface node ⟨*num*⟩, where the number is a positive integer or `N`. The last node is always defined as `N`.

`\oenodeIn{`⟨*comp*⟩`}`

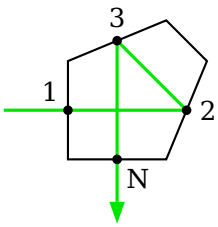
The interface node ⟨*1*⟩ is the input node and should always be accessed via `\oenodeIn`.

`\oenodeOut{`⟨*comp*⟩`}`

The interface node ⟨*N*⟩ is the output node and should always be accessed via `\oenodeOut`.

"Input" and "output" can define only relative orientations. The input node is by definition the node to which a beam coming from the reference node `\oenodeRefA` is connected to. Correspondingly is the definition of the output node. This definition breaks down only for fiber couplers (see Sec. 5.10.1) and beamsplitter (4.13).

\oenodeIn is equivalent to \oenodeIfc{1} and \oenodeOut is equivalent to \oenodeIfc{N}. You should use the \oenodeIfc macro only to access the nodes which are not the input and output nodes as for those explicit macros are provided.




```
\begin{pspicture}(2.8,3)
  \pentaprism[pentaprismsize=1.3](0,1.5)(1.5,1.5)(1.5,0)
  \drawbeam[arrows=->, arrowinset=0,
     arrowscale=1.5](0,1.5){}(1.5,0)
  \psdot(\oenodeIn{})\uput[135](\oenodeIn{}){1}
  \psdot(\oenodeIfc{2}{})\uput[0](\oenodeIfc{2}{}){2}
  \psdot(\oenodeIfc{3}{})\uput[90](\oenodeIfc{3}{}){3}
  \psdot(\oenodeOut{})\uput[-45](\oenodeOut{}){N}
\end{pspicture}
```

See Sec. 11.2.2 for a complete list of all components with their interface nodes.

7.7. Rotation reference node

\oenodeRotref{⟨*comp*⟩}

The node around which a component is rotated by angle. The position is defined with the rotateref parameter and can take the same values as extnode. See Sec. 11.2.1 for possible rotation reference nodes of all components.

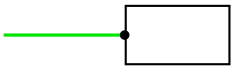

```
\begin{pspicture}(0,0.1)(12,1.8)
  \optbox[angle=20](0.5,1)(4.5,1)
  \psdot(\oenodeRotref{})\uput[-90](\oenodeRotref{}){Rotref}
  \optbox[angle=20, rotateref=bl](7.5,1)(11.5,1)
  \psdot(\oenodeRotref{})\uput[-90](\oenodeRotref{}){Rotref}
\end{pspicture}
```

7.8. Beam nodes

The end points of a recent \drawbeam or \drawwidebeam command can be reused if savebeampoints is set.

\oenodeBeam{⟨num⟩}

Access the end point of a preceeding \drawbeam command. If ⟨num⟩ is left empty it defaults to 1 which usually refers to the last beam ray. However, generally it depends on the settings of savebeampoints and the choice of ⟨num⟩ which beam you refer to.

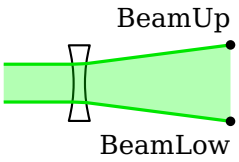


```
\begin{pspicture}(0,0)(3,2)
  \pnode(0,1){A}\pnode(1.6,1){B}
  \optbox[position=end](A)(B)
  \drawbeam[beaminsidelast=false](A){1}
  \psdot(\oenodeBeam{})
\end{pspicture}
```

\oenodeBeamUp{⟨num⟩}

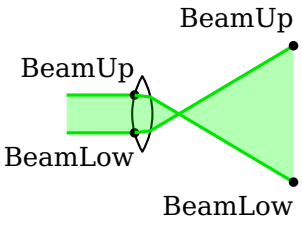
\oenodeBeamLow{⟨num⟩}

Access the upper or lower end point of a preceeding \drawwidebeam command. If ⟨num⟩ is left empty it defaults to 1, which usually refers to the last beam ray. However, generally it depends on the settings of savebeampoints and the choice of ⟨num⟩ which beam you refer to.



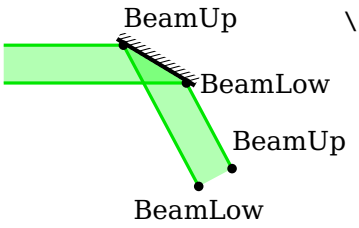

```
\begin{pspicture}(0,0)(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \lens[abspos=1, lensradius=-2](A)(B)
  \addtopsstyle{Beam}{%
    fillstyle=solid, fillcolor=green, opacity=0.3}
  \drawwidebeam[beamwidth=0.5](A){}(B)
\end{pspicture}
```

Note, that “upper” and “lower” is defined with respect to the propagation direction of the beam. A marginal beam which started as “upper” can change to “lower” if it crosses the other marginal beam. This is shown in the following example, where the marginal rays exchange their roles:




```
\begin{pspicture}(0,0)(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \lens[abspos=1, n=2](A)(B)
  \addtopsstyle{Beam}{%
    fillstyle=solid, fillcolor=green, opacity=0.3}
  \drawwidebeam[beamwidth=0.5](A){}
  \drawwidebeam[loadbeam, beaminsidefirst]{}(B)
\end{pspicture}
```

After a reflection the roles can also be exchanged.



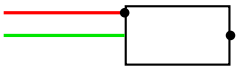

```
\begin{pspicture}(0,-1)(4,2)
  \pnode(0,1.5){A}\pnode(2,1.5){B}\pnode(2.8,0){C}
  \mirror[mirrorwidth=1.2, mirrortype=extended](A)(B)(C)
  \addtopsstyle{Beam}{%
    fillstyle=solid, fillcolor=green, opacity=0.3}
  \drawwidebeam[beamwidth=0.5](A){}
  \drawwidebeam[loadbeam]{}(C)
\end{pspicture}
```

Note, that the orientation of “upper” and “lower” changed with version 3.3, for more details see Sec. 11.3.1.

The beam nodes can be accessed only within the `pspicture` environment they were defined in. If you want to use one of these nodes from outside, you must explicitely redefine them beforehand:

```
\pnode(\oenodeBeam{}){MyBeamNode}
```

The end points are also affected by `stopinside`. See Sec. 8.5 for further details.



```
\begin{pspicture}(0,0)(3,2)
  \pnode(0,1){A}\pnode(1.6,1){B}
  \optbox[position=end](A)(B)
  \psset{beaminsidelast=false}
  \drawbeam[linecolor=red, beampos=0.3](A){1}
  \psdot(\oenodeBeam{})
  \drawbeam[stopinside](A){1}
  \psdot(\oenodeBeam{})
\end{pspicture}
```

7.9. Beam vector

In addition to the beam nodes (see Sec. 7.8) also the vectors of the last saved beams (see `savebeam`) can be access on Postscript level.

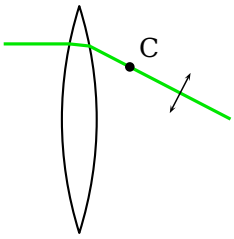
These are again separated in single, upper and lower beams. ⟨*num*⟩ has the same meaning as for `\oenodeBeam`.

`\oeBeamVec`{⟨*num*⟩}

`\oeBeamVecUp`{⟨*num*⟩}

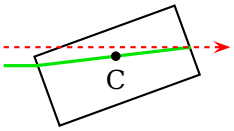
`\oeBeamVecLow`{⟨*num*⟩}

A possible application is marking of the polarization of a beam. In the following example we define a node ⟨C⟩ which is shifted with \nodexn from the beam end point along the beam trajectory.



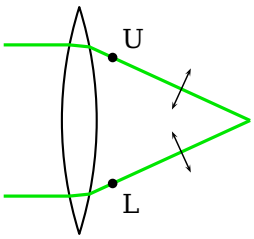

```
\begin{pspicture}(3,3)
  \pnode(0,1.5){A}\pnode(3,1.5){B}
  \lens[abspos=1, lens=5 5 3, n=2.1](A)(B)
  \optplane(B)
  \drawbeam[beampos=1](A){1-2}
  \nodexn{(\oenodeBeam{})-1.5(!\oeBeamVec{})}{C}
  \polarization(C)(\oenodeBeam{})
\end{pspicture}
```

This examples shows that the last incoming beam vector is used, instead of loadbeam, where the outgoing beam vector is used. The direction of the outgoing beam is marked red.




```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \optbox[optboxsize=2 1, angle=20](A)(B)
  \drawbeam[beaminsidelast, beamalign=abs](A){1}
  \nodexn{(\oenodeBeam{})-(!\oeBeamVec{})}{C}
\end{pspicture}
```

This can be applied to the upper and lower beams as well.

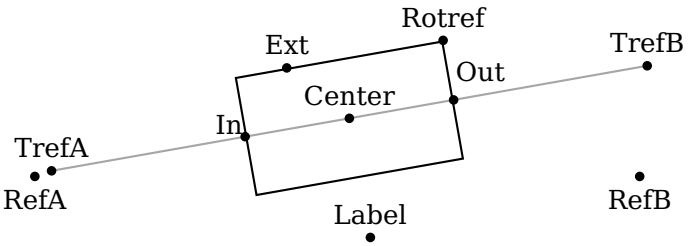



```
\begin{pspicture}(3,3)
  \pnode(0,1.5){A}\pnode(3,1.5){B}
  \lens[abspos=1, lens=5 5 3, n=2.0](A)(B)
  \optplane[angle=90](B)
  \drawwidebeam[beamwidth=2](A){1-2}
  \nodexn{(\oenodeBeamUp{})-2(!\oeBeamVecUp{})}{U}
  \nodexn{(\oenodeBeamLow{})-2(!\oeBeamVecLow{})}{L}
  \polarization(U)(\oenodeBeamUp{})
  \polarization(L)(\oenodeBeamLow{})
\end{pspicture}
```

For another example see Ex. 10.

7.10. Node overview

This is an overview of all available component nodes.




```
\begin{pspicture}(0.3,0.1)(4.8,1.6)
  \pnode(0.5,0.5){A}\pnode(4.5,0.5){B}
  \optbox[compshift=0.5, angle=10, rotateref=tr, extnode={-0.5,1}](A)(B)
\end{pspicture}
```

8. Connecting components

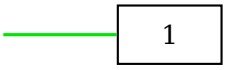
The pst-optexp package provides several methods for automatic and manual beam and fiber drawing.

- Sec. 8.1 describes how to access objects and nodes for use with connections.
- Beam drawing in general and the behavior of single beams is explained in Sec. 8.2.
- Sec. 8.3 extends this description of beam drawing to wide beams.
- Sec. 8.4 shows how errors in beam connections (missed interfaces etc.) are handled.
- Sec. 8.5 explains how customized, piecewise defined beam paths are constructed.
- Manual and automatic fiber connections are described in Sec. 8.6.
- Sec. 8.7 introduces the concept of front and back layer for the drawings.

8.1. Accessing components

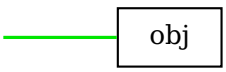
All macros for connections can take either an ID, a compname identifier (see Sec. 7.1) or PSTricks nodes as arguments, which are denoted with $\langle obj_1 \rangle$, $\langle obj_2 \rangle$, ... To distinguish between nodes and components, nodes must be either enclosed in parenthesis within the brackets, or enclosed only in parenthesis: \drawbeam{(node)}{comp}, or \drawbeam(node){comp}.

1. All components can always be accessed via their ID.



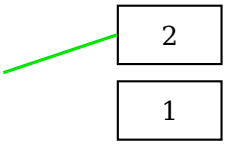

```
\begin{pspicture}(3,1.5)
  \pnode(0,1){A}\pnode(1.5,1){B}
  \optbox[position=end, labeloffset=0](A)(B){1}
  \drawbeam(A){1}
\end{pspicture}
```

2. A component which has been assigned a name with compname can also be accessed via this instead of its ID.




```
\begin{pspicture}(3,2)
  \psset{position=end, labeloffset=0}
  \optbox[compname=obj](0,1)(1.5,1){obj}
  \drawbeam(0,1){obj}
\end{pspicture}
```

3. If the argument is left empty, use the last component defined.




```
\begin{pspicture}(3,2)
  \psset{position=end, labeloffset=0}
  \optbox(0,0.5)(1.5,0.5){1}
  \optbox(0,1.5)(1.5,1.5){2}
  \drawbeam(0,1){}
\end{pspicture}
```

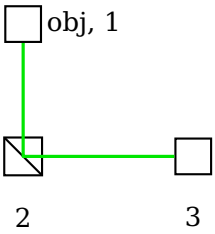
4. For \drawbeam and \drawwidebeam you can also specify a number range. Valid range specifications are

   x-y From x to y, if x is greater than y, the numbers are decremented. If the whole range is not defined, an error is raised.

   x- From x to the last component.

   -y From the first component to y. If y is outside the valid range, a warning is raised.

   - Connect all components.




```
\begin{pspicture}(3,3)
  \psset{position=end, optboxsize=0.5 0.5, bssize=0.5}
  \optbox[compname=obj](0.5,0.5)(0.5,2.5){obj, 1}
  \beamsplitter(0.5,2.5)(0.5,1)(2.5,1){2}
  \optbox(0.5,1)(2.5,1){3}
  \drawbeam{1-3}
\end{pspicture}
```

8.2. Drawing beams

The package provides several ways to connect components and nodes with beams, which can be either single rays or wide beams. Most parameters have the same effect on both single and wide beams so most examples will be shown for single beams only. For options specific to wide beams please see Sec. 8.3.

`\drawbeam`[⟨*options*⟩]{⟨*obj*$_1$⟩}{⟨*obj*$_2$⟩}...

`\drawwidebeam`[⟨*options*⟩]{⟨*obj*$_1$⟩}{⟨*obj*$_2$⟩}...

These are the two macros for beam connections; they can take a variable number of arguments (minimum of two) which can be either an ID, a `compname`, or a PSTricks nodes. To distinguish between nodes and components, nodes must be either enclosed in parenthesis within the brackets, or enclosed only in parenthesis: `\drawbeam{(node)}{comp}`, or `\drawbeam(node){comp}`.

8.2.1. Raytracing

The beam connection macros support two modes of tracing the beam across the optical components: one is to use refractive indices and Snell's law to determine the beam path. The other mode is to connect the components only, regardless of their optical properties.

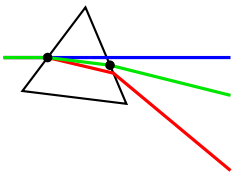
You must always keep in mind, that this is a package for *sketching* experimental setups. Therefore, we do not provide a comprehensive raytracing framwork, because deviations from the actual physical path are often desired for sketches: beam angles and divergences should be in many cases more extreme than in the real setup in order to highlight certain aspects. You should also consider the refractive index (see Sec. 8.2.2) only as a tool to optimize the visual effects of the beam path, and not stick with its physically correct values.

`raytrace`=`true`, `false` default: `true`

Chooses the mode of tracing the beam path, `true` uses the refractive index ("raytracing"), `false` selects component connection ("connect" mode).

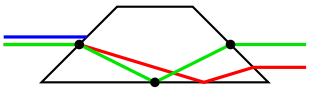
In most cases, the "connect" mode is equivalent to "raytracing" with `n`=1, except for `\optprism` and `\doveprism`, where the interface nodes (Sec. 7.6) are simply connected. See the following two example for the differences between the tracing modes using these two components. The interface nodes are highlighted for clarity.

The raytraced beam with `n`=1 (blue) passes the component without changing its direction, the refraction for `n`=1.5 is calculated (red), and the "connect" beam (green) just passes through the two interface nodes.



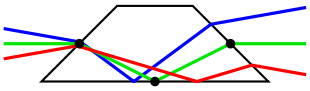
```
\begin{pspicture}(3,2.2)
  \pnode(0,1.5){A}\pnode(1,1.5){B}\pnode(3,1){C}
  \optprism[prismsize=1.2](A)(B)(C)
  \optplane(C)
  \drawbeam[n=1, linecolor=blue](A){1}{2}
  \drawbeam[n=1.5, linecolor=red](A){1}{2}
  \drawbeam[raytrace=false, linecolor=DGreen](A){1}{2}
\end{pspicture}
```

For the \doveprism the same applies as for the \optprism, but the raytraced beam with n=1 (blue) misses the second interface because it is not refracted and stops therefore at the first interface (see Sec. 8.4). Again, the green “connect” ray passes through all interface nodes.

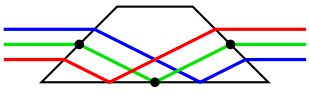


```
\begin{pspicture}(4,2)
  \pnode(0,1){A}\pnode(4,1){B}
  \doveprism[doveprismsize=1](A)(B)
  \drawbeam[n=1, linecolor=blue, beampos=0.1](A){1}(B)
  \drawbeam[n=1.5, linecolor=red](A){1}(B)
  \drawbeam[raytrace=false, linecolor=DGreen](A){1}(B)
\end{pspicture}
```

The “connect” mode also supports changing the initial conditions (8.2.3).



```
\begin{pspicture}(3.7,2)
  \pnode(0,1){A}\pnode(4,1){B}
  \doveprism[doveprismsize=1](A)(B)
  \psset[optexp]{raytrace=false}
  \drawbeam[linecolor=blue, beampos=0.2,
     beamangle=-10](A){1}(B)
  \drawbeam[raytrace=false, linecolor=DGreen](A){1}(B)
  \drawbeam[linecolor=red, beampos=-0.2, beamangle=10](A){1}(B)
\end{pspicture}
```

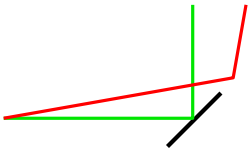


```
\begin{pspicture}(3.7,2)
  \pnode(0,1){A}\pnode(4,1){B}
  \doveprism[doveprismsize=1](A)(B)
  \psset[optexp]{raytrace=false}
  \drawbeam[linecolor=blue, beampos=0.2](A){1}(B)
  \drawbeam[linecolor=DGreen](A){1}(B)
  \drawbeam[linecolor=red, beampos=-0.2](A){1}(B)
\end{pspicture}
```

useNA=true, false default: true

The beam path is interrupted if an interface is missed. In some cases, e.g. to estimate the correct refractive index for a ray trace, it can be useful if the

numerical aperture of the components is not considered. The red beam in the following example is drawn, although it hits the mirror outside the drawn mirror surface.



```
\begin{pspicture}(3,2)
  \pnode(0,0.5){A}\pnode(2.5,0.5){B}\pnode(2.5,2){C}
  \mirror(A)(B)(C)
  \psset{useNA=false}
  \drawbeam(A){}(C)
  \drawbeam[linecolor=red, beamangle=10](A){}(C)
\end{pspicture}
```

8.2.2. Refractive index

The raytracing depends on the refractive index, which may be set for each component separately. The refractive index is always relative to the background which has a constant value for the whole sketch.

n=⟨*code*⟩ default: `1.5`

Sets the relative refractive index respect to the background. This parameter has two scopes, you can either set the refractive index of each component or of each beam ray:

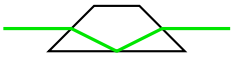
1. Using `n` as global parameter or component argument sets a fixed refractive index for each affected component. Without further actions, all beam paths experience this refractive index.
2. As parameter for `\drawbeam` or `\drawwidebeam` you can overwrite or even change the refractive index of the components for specific beam paths. This can be very useful e.g. to simulate chromatic dispersion. To set this globally for all beams you must add the respective parameter definition to the `Beam` style because global definitions of `n` affect only the components and not the beams.

The ⟨*code*⟩ can in general be any Postscript code which evaluates to a number, e.g. `1.5`, or `5 sqrt`. If the ⟨*code*⟩ starts with a `*`, it will be interpreted as an algebraic expression[1], e.g. `*sqrt(5)` is equivalent to `5 sqrt`. Inside the ⟨*code*⟩ you can access the predefined refractive index of the component via `n` which allows you to change the index relative to its original value. Use e.g. `n=n 1.01 mul` to change `n` by one percent.

[1] http://mirror.ctan.org/graphics/pstricks/base/doc/pst-news08.pdf

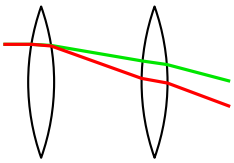
In the following you find several examples about the different aspects of using `n`.

1. Set a fixed refractive index for the component. Interestingly, for the default shape of the `\doveprism` a refractive index of $\sqrt{5}$ gives the ideal output direction which is the same as the input direction.



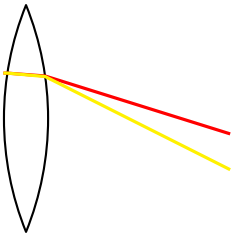
```
\begin{pspicture}(3,1.5)
  \pnode(0,1){A}\pnode(3,1){B}
  \doveprism[n=*sqrt(5)](A)(B)
  \drawbeam(A){}(B)
\end{pspicture}
```

2. In this example one beam uses the default refractive index of the components (green), the other one overwrites it with a value of 2 (red).



```
\begin{pspicture}(3,3)
  \pnode(0,1){A}\pnode(3,1){B}
  \optplane(A)
  \psset{lens=3 3 2, n=1.5}
  \lens[abspos=0.5](A)(B)
  \lens[abspos=2](A)(B)
  \optplane(B)
  \addtopsstyle{Beam}{beampos=0.5}
  \drawbeam{-}
  \drawbeam[linecolor=red, n=2]{-}
\end{pspicture}
```

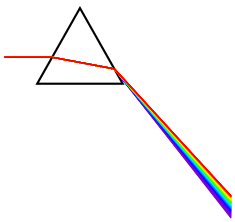
3. Now, the red beam uses the default refractive index of the component, the yellow beam adds `0.5` to the default index.



```
\begin{pspicture}(3,3)
  \pnode(0,1.5){A}\pnode(3,1.5){B}
  \lens[lens=4 4 3, n=2, abspos=0.3](A)(B)
  \optplane[compname=Plane](B)
  \psset{beampos=0.6}
  \drawbeam[linecolor=red](A){1}{Plane}
  \drawbeam[linecolor=yellow, n=*(n+0.5)](A){1}{Plane}
\end{pspicture}
```

4. Dynamically changing the refractive index is very useful to emulate chromatic dispersion. Here, we sketch the dispersion of a prism by calculating the refractive index with Sellmaier's equation[2] for different wavelengths.

[2] http://en.wikipedia.org/wiki/Sellmeier_equation



```
\begin{pspicture}(3,3)
  \pnode(-1,0){A}\pnode(1,2.2){B}\pnode(3,0){C}
  \optplane(0,2.15)
  \optprism[prismalign=center, prismangle=59](A)(B)(C)
  \optplane(C)
  \definecolor[ps]{bl}{rgb}{%
    tx@addDict begin Red Green Blue end}%
  \addtopsstyle{Beam}{linecolor=bl,
      linewidth=0.4\pslinewidth, beamalign=abs}
  \multido{\i=0+1}{60}{%
    \pstVerb{%
      \i\space 650 400 sub 59 div mul 400 add
      tx@addDict begin wavelengthToRGB end }%
    \drawbeam[n=\i\space 650 400 sub 59 div mul 400 add
        Sellmaier]{-}%
  }%
\end{pspicture}
```

5. Using the numerical aperture of a pinhole as spectral filter.

```
\begin{pspicture}(0.3,0)(3.3,5.2)
  \pnode(0.3,1){A}\pnode(2,1){B}\pnode(3,2){C}\pnode(2.5,5){D}
  \optplane(A)
  \optgrating[reverse](A)(B)(C)
  \pinhole[phwidth=-0.1, innerheight=0.03, abspos=0.5](D)(B)
  \optplate[position=end, plateheight=1.5](B)(D)
  \definecolor[ps]{bl}{rgb}{%
    tx@addDict begin Red Green Blue end}%
  \drawbeam[linecolor=red]{1-2}
  \addtopsstyle{Beam}{linecolor=bl,
      linewidth=0.4\pslinewidth}
  \multido{\i=0+1}{60}{%
    \pstVerb{%
      \i\space 650 400 sub 59 div mul 400 add
      tx@addDict begin wavelengthToRGB end }%
    \drawbeam[beamangle=\i\space 0.1 mul 3 sub]{2-}%
  }%
\end{pspicture}
```

refractiveindex

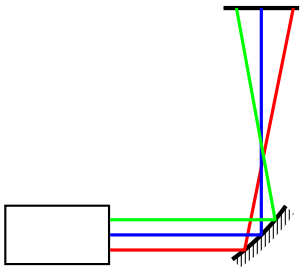
=⟨*code*⟩

This parameter is deprecated since version 3.0, use n instead.

8.2.3. Initial conditions

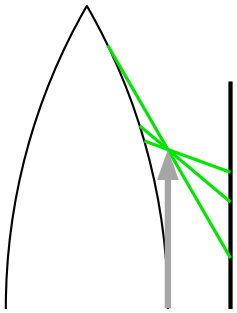
beampos=[⟨*x*⟩]⟨*y*⟩ default: 0

This is the start position (⟨*x*⟩, ⟨*y*⟩) of the beam at the first interface. Both values are of ⟨*psnum*⟩ type. If a single number is given, the x-coordinate is set to 0.



```
\begin{pspicture}(4,3.8)
  \pnode(1.5,0.5){A}\pnode(3.5,0.5){B}\pnode(3.5,3.5){C}
  \optbox[position=end](B)(A)
  \mirror[mirrortype=extended, mirrorradius=3](A)(B)(C)
  \optplate[position=end](B)(C)
  \drawbeam[linecolor=red, beampos=-0.2]{1-3}
  \drawbeam[linecolor=blue, beampos=0]{1-3}
  \drawbeam[linecolor=green, beampos=0.2]{1-3}
\end{pspicture}
```

The start position is relative to the respective interface node, as it is illustrated in the following example. The beam start position is fixed, but depending on e.g. the beamangle, the actual intersection with the curved interface is at a different position.

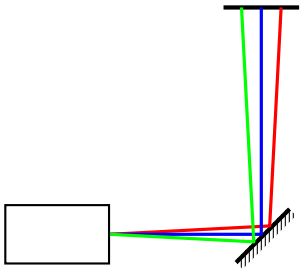


```
\begin{pspicture*}(0,2)(3.05,6)
  \lens[lens=8 8 8, abspos=0.6](0.5,2)(3,2)
  \optplate[position=1, plateheight=6](0.5,2)(3,2)
  \psset{beampos=2.1}
  \drawbeam[beamangle=-60]{1}{2}
  \drawbeam[beamangle=-40]{1}{2}
  \drawbeam[beamangle=-20]{1}{2}
\end{pspicture*}
```

Which interface the beam starts at is determined by the options beaminsidefirst and startinside.

beamangle=⟨*psnum*⟩ default: 0

This is the start angle of the beam. Usually, it is relative to the connection between the first two components, use beamalign to change this.

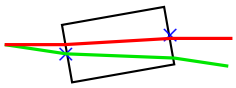


```
\begin{pspicture}(4,4)
  \pnode(1.5,0.5){A}\pnode(3.5,0.5){B}\pnode(3.5,3.5){C}
  \optbox[position=end](B)(A)
  \mirror[mirrortype=extended](A)(B)(C)
  \optplate[position=end](B)(C)
  \drawbeam[linecolor=red, beamangle=3]{-3}
  \drawbeam[linecolor=blue, beamangle=0]{1-}
  \drawbeam[linecolor=green, beamangle=-3]{-}
\end{pspicture}
```

beamalign=rel, relative, abs, absolute default: relative

Select whether the beamangle is relative to the connection between the first two components, or if it is an absolute angle.

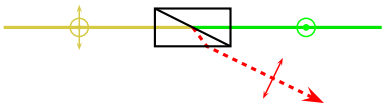
A typical situation where an absolute angle is helpful is with rotated or shifted components. By rotating or shifting a component also its interface nodes are rotated (see Sec. 3.3) which are then used to determine the input angle for relative alignment. Accordingly, in the following example, the green beam with the relative alignment is not horizontal.



```
\begin{pspicture}(3,1.5)
  \pnode(0,0.75){A}\pnode(3,0.75){B}
  \optbox[angle=10, showifcnodes](A)(B)
  \drawbeam(A){}(B)
  \drawbeam[linecolor=red, beamalign=abs](A){}(B)
\end{pspicture}
```

beampathskip=⟨*num*⟩ default: 0

Skip ⟨*num*⟩ segments at the beginning of the beam path.



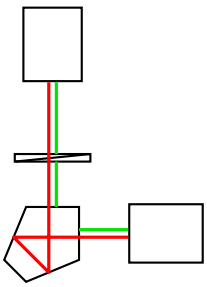
```
\begin{pspicture}(5,2)
\pnode(0,1){A}\pnode(5,1){B}\pnode(2,0){C}
\begin{optexp}
  \glanthompson(A)(B)
  \optplane(B)\optplane[angle=90](C)
  \polarization[abspos=1, poltype=misc, linecolor=yellow!80!black](A)(B)
  \drawbeam[beaminsidelast, linecolor=yellow!80!black](A){1}
  \drawbeam[beampathskip=2](A){1-2}
  \drawbeam[beampathskip=2, savebeam, linestyle=dashed, dash=2pt 2pt,
      linecolor=red,arrowscale=1.5, arrows=->](A){1}{3}
\end{optexp}
\nodexn{(\oenodeBeam{})-1.5(!\oeBeamVec{})}{D}
\polarization[poltype=parallel, linecolor=red](D)(\oenodeBeam{})
\polarization[abspos=4, poltype=perp, linecolor=green, dotscale=1.5](A)(B)
\end{pspicture}
```

8.2.4. Internal beam path

A beam path is composed of rays between components and rays inside the components. Depending on some options these internal beams can be drawn or not. Generally, the first, the last, and all other components are treated separately.

`beaminside`=`true`, `false` default: `true`

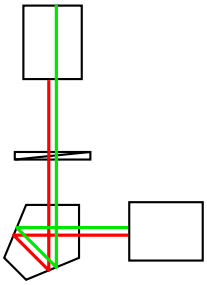
Draw the internal beams in all intermediate components, the first and last components are not affected.



```
\begin{pspicture}(0.3,0.3)(3,4)
  \psset{optboxwidth=1}
  \optbox[position=end](1,1)(1,3)
  \optretplate(1,1)(1,3)
  \pentaprism(1,3)(1,1)(2,1)
  \optbox[position=end](1,1)(2,1)
  \drawbeam[beampos=-0.05, linecolor=red]{-}
  \drawbeam[beampos=0.05, beaminside=false]{-}
\end{pspicture}
```

`beaminsidefirst`=`true`, `false` default: `false`

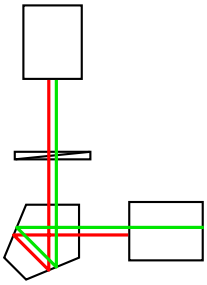
Draw the internal beams of the first component.



```
\begin{pspicture}(0.3,0.3)(3,4)
  \psset{optboxwidth=1}
  \optbox[position=end](1,1)(1,3)
  \optretplate(1,1)(1,3)
  \pentaprism(1,3)(1,1)(2,1)
  \optbox[position=end](1,1)(2,1)
  \drawbeam[beampos=-0.05, linecolor=red]{-}
  \drawbeam[beampos=0.05, beaminsidefirst]{-}
\end{pspicture}
```

beaminsidelast=true, false default: false

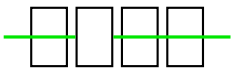
Draw the internal beams of the last component.



```
\begin{pspicture}(0.3,0.3)(3,4)
  \psset{optboxwidth=1}
  \optbox[position=end](1,1)(1,3)
  \optretplate(1,1)(1,3)
  \pentaprism(1,3)(1,1)(2,1)
  \optbox[position=end](1,1)(2,1)
  \drawbeam[beampos=-0.05, linecolor=red]{-}
  \drawbeam[beampos=0.05, beaminsidelast]{-}
\end{pspicture}
```

allowbeaminside=true, false default: true

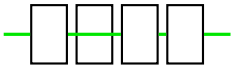
Define whether a beam is drawn inside a component. The option affects components only and overwrites all settings of beaminside, beaminsidefirst, and beaminsidelast if set to false. With this you can forbid any internal beams for components where it does not make any sense (e.g. \optfilter and \optdiode).



```
\begin{pspicture}(3,2)
  \psset{optboxwidth=0.5}
  \pnode(0,1){A}\pnode(3,1){B}
  \optbox[position=0.2](A)(B)
  \optbox[position=0.4, allowbeaminside=false](A)(B)
  \optbox[position=0.6](A)(B)
  \optbox[position=0.8](A)(B)
  \drawbeam(A){-}(B)
\end{pspicture}
```

forcebeaminside=true, false default: false

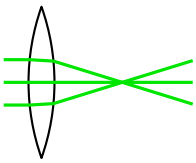
Forces to draw a beam inside a component. The option affects components only and overwrites all settings of allowbeaminside, beaminside, beaminsidefirst, and beaminsidelast if set to true. With this you can force an internal beams for single components within a beam path.



```
\begin{pspicture}(3,2)
  \psset{optboxwidth=0.5}
  \pnode(0,1){A}\pnode(3,1){B}
  \optbox[position=0.2](A)(B)
  \optbox[position=0.4, forcebeaminside](A)(B)
  \optbox[position=0.6](A)(B)
  \optbox[position=0.8](A)(B)
  \drawbeam[beaminside=false](A){-}(B)
\end{pspicture}
```

8.2.5. Connecting with nodes

As mentioned earlier in this chapter, beams can connect components also with nodes. For raytracing you need in general a plane, which imposes some problems when connecting with nodes. This is handled such, that a temporary plane is constructed, which goes through the specified node and is perpendicular to the connection between the node and the preceeding component. If a beam starts with a node, the orientation is determined by the node and the following component.

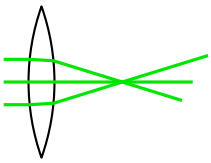


```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(2.5,1){B}
  \lens[lens=3 3 2, n=2.5, abspos=0.5](A)(B)
  \multido{\r=-0.3+0.3}{3}{%
    \drawbeam[beampos=\r](A){}(B)}
\end{pspicture}
```

This is very handy in many cases, but does not allow to rotate the node planes and can give again problems with shifted and rotated components (see Sec. 3.3). For more flexibility, the package provides an “invisible” plane which can be used for this purpose.

`\optplane`(⟨*center*⟩)

The plane is defined by a single node only, the rotation around this node is defined with `angle`. The angle is with respect to the default alignment, which is vertical.



```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(2.5,1){B}
  \lens[lens=3 3 2, n=2.5, abspos=0.5](A)(B)
  \optplane[angle=-30](B)
  \multido{\r=-0.3+0.3}{3}{%
    \drawbeam[beampos=\r](A){-}}
\end{pspicture}
```

8.2.6. Automatic connection

Beam can also be drawn at definition time of a component, which can be handy, but allows only very elemental beams.

`beam`=`true`, `false` default: `false`

Can be used as option for every component, is equivalent to the statement

```
\drawbeam[raytrace=false](\oenodeRefA{}){}(\oenodeRefB{})
```

which draws a beam from one reference node via the component to the other reference node.

`conn`=⟨*string*⟩

This option is maintained for backward compatibility only and will be removed in future versions. See Sec. 11.3.2 for further information.

8.2.7. Beam appearance

`Beam` ⟨*psstyle*⟩ default: `linecolor=green!90!black, linejoin=1`

The appearance of beams is controlled with this style. The optional argument of `\draw*beam` can also be used, these options can overwrite settings from the `Beam` style.

`addtoBeam`=⟨*list*⟩

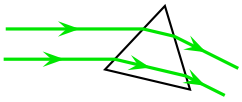
The `Beam` style is extended by locally adding the parameters contained in ⟨*list*⟩. The ⟨*list*⟩ must be surrounded by curly braces.

`newBeam`=⟨*list*⟩

Similar to `addtoBeam`, but an existing `Beam` style is overwritten with the new parameter set.

ArrowInsideMinLength=⟨*psnum*⟩
default: 0.2

\draw*beam uses also the option ArrowInside from pstricks-add. In this case the arrows would be drawn for every beam segment, which is often undesired for the internal, short beam paths. ArrowInsideMinLength sets a minimum segment length, below which no internal arrows are drawn. In the following example the value is selected such that only the lower internal beam has an arrow (see also Ex. 10).

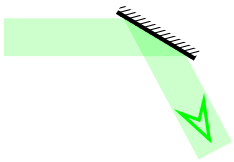


```
\begin{pspicture}(3,2)
  \optprism(0,1)(2,1)(3,0.5)
  \drawwidebeam[beamwidth=0.4, ArrowInside=->,
     ArrowInsideMinLength=0.5, arrowscale=1.5,
     raytrace=false](0,1){}(3,0.5)
\end{pspicture}
```

ArrowInsideMaxLength=⟨*psnum*⟩
default: -1

Like ArrowInsideMaxLength, but sets the maximum length of a segment for which the arrows are drawn. If the value is negative, no upper limit is defined.

ArrowInside can be used also without a linestyle.



```
\begin{pspicture}(0,0)(3.5,2)
  \pnode(0,1.5){A}\pnode(2,1.5){B}\pnode(2.8,0){C}
  \mirror[mirrorwidth=1.2, mirrortype=extended](A)(B)(C)
  \newpsstyle{Beam}{linecolor=green, linestyle=none,
     fillstyle=solid, fillcolor=green, opacity=0.2}
  \drawwidebeam[beamwidth=0.5](A){}(C)
  \drawbeam[ArrowInside=->, arrowscale=2.5, ArrowFill=false,
     linewidth=1.5\pslinewidth, ArrowInsidePos=0.9,
     ArrowInsideMaxLength=1.8](A){}(C)
\end{pspicture}
```

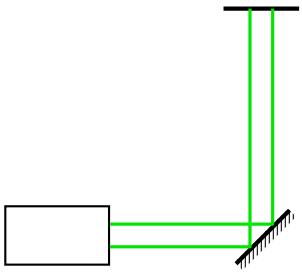
8.3. Drawing wide beams

All information about single beams (see Sec. 8.2) apply to wide beams as well. Here, you find all additional information which apply to wide beams only.

\drawwidebeam[⟨*options*⟩]{⟨*obj*$_1$⟩}{⟨*obj*$_2$⟩}...

beamwidth=⟨*psnum*⟩ default: 0

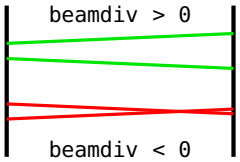
The initial beam width, measured at the initial position (defined with beampos) perpendicular to the beam direction.



```
\begin{pspicture}(4,4)
  \pnode(1.5,0.5){A}\pnode(3.5,0.5){B}\pnode(3.5,3.5){C}
  \optbox[position=end](B)(A)
  \mirror[mirrortype=extended](A)(B)(C)
  \optplate[position=end](B)(C)
  \drawwidebeam[beamwidth=0.3]{-}
\end{pspicture}
```

beamdiv=⟨*psnum*⟩ default: 0

The total beam divergence at the starting position, given in degree. Positive values give an expanding beam, negative values a contracting beam.



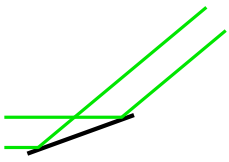

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \psset{plateheight=2}
  \optplate[position=start](A)(B)
  \optplate[position=end](A)(B)
  \addtopsstyle{Beam}{beamwidth=0.2}
  \drawwidebeam[beampos=0.4, beamdiv=5]{-}
  \drawwidebeam[beampos=-0.4, beamdiv=-5, linecolor=red]{-}
\end{pspicture}
```

8.3.1. Beam appearance

The appearance of wide beams can be controlled like that of single beams with the optional argument or the Beam style (see Sec. 8.2.7).

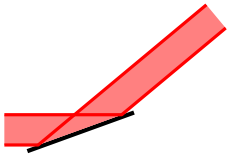
Wide beams have two marginal rays, which are drawn according to the line settings, and may be filled.

1. Only marginal rays.



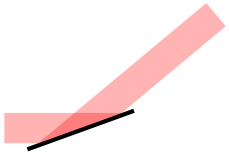
```
\begin{pspicture}(3,2)
  \pnode(0,0.5){A}\pnode(1,0.5){B}\pnode(2.8,2){C}
  \mirror[mirrorwidth=1.5](A)(B)(C)
  \drawwidebeam[beamwidth=0.4](A){}(C)
\end{pspicture}
```

2. Marginal rays, and with fill style.



```
\begin{pspicture}(3,2)
  \pnode(0,0.5){A}\pnode(1,0.5){B}\pnode(2.8,2){C}
  \mirror[mirrorwidth=1.5](A)(B)(C)
  \addtopsstyle{Beam}{fillstyle=solid,
      fillcolor=red!50!white, linecolor=red}
  \drawwidebeam[beamwidth=0.4](A){}(C)
\end{pspicture}
```

3. Only with fill style.

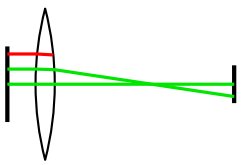


```
\begin{pspicture}(3,2)
  \pnode(0,0.5){A}\pnode(1,0.5){B}\pnode(2.8,2){C}
  \mirror[mirrorwidth=1.5](A)(B)(C)
  \addtopsstyle{Beam}{fillstyle=solid, fillcolor=red,
      opacity=0.3, linestyle=none}
  \drawwidebeam[beamwidth=0.4](A){}(C)
\end{pspicture}
```

8.4. Error handling

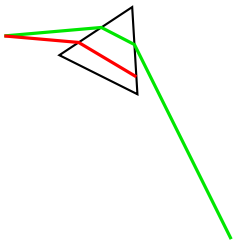
All beam are calculated and drawn directly in Postscript, so that a comprehensive error handling at compilation time is not possible. If a situation arises which is not supported by the drawing algorithm, the beam path is truncated at this point. This happens in the following cases:

1. A beam misses an interface. The red beam in the following example misses the last plane and is truncated at the last valid interface.



```
\begin{pspicture}(3,2.7)
  \pnode(0,1){A}\pnode(3,1){B}
  \optplate[position=start](A)(B)
  \lens[lens=4 4 2, abspos=0.5, n=2.5](A)(B)
  \optplate[position=end, plateheight=0.5](A)(B)
  \drawbeam{-}
  \drawbeam[beampos=0.2]{-}
  \drawbeam[beampos=0.4, linecolor=red]{-}
\end{pspicture}
```

2. When total internal reflection occurs at an interface which is not set up for this. The red beam in the following example would be reflected, but this is not supported and the beam is truncated at the interface where the total internal reflection occurs.



```
\begin{pspicture}(3,2.5)
  \pnode(0,2){A}\pnode(3,0){B}
  \optplane(A)
  \optprism[n=1.8](A)([Xnodesep=1.5]A)(B)
  \optplane(B)
  \drawbeam[beamangle=5]{-}
  \drawbeam[beamangle=-5, linecolor=red]{-}
\end{pspicture}
```

These conditions apply to wide beams (see Sec. 8.3) as well, the beam path is interrupted as soon as one of the marginal rays has an error.

`pswarning`=`true`, `false` default: `false`

To simplify debugging, you can print a message to standard output if a beam misses an interface or if total internal reflection occured at some point.

8.5. Custom beams

With an appropriate choice of refractive index and component parameters (e.g. lens curvature) all kind of beam paths could be realised although it might be very cumbersome. To simplify this we provide the possibility to use the end points of one beam path as starting points for the next beam which allows piecewise definition of the beam divergence, angle, and direction.

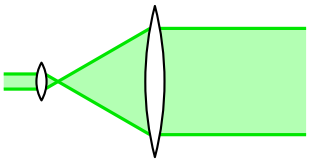
`savebeampoints`=`true`, `false`, ⟨*int*⟩ default: `true`

Saves the end points of a `\draw*beam` macro, previous points are overwritten. `\drawbeam` and `\drawwidebeam` are treated separately. If a number ⟨*int*⟩ > 0 is given, you can save as many end points as you like, but in most use cases enabling and disabling this option is sufficient.

`loadbeampoints`=`true`, `false`, ⟨*int*⟩ default: `false`

Use the end points of a previous beam as starting points for the next beam. If no number ⟨*int*⟩ is specified, `1` is used.

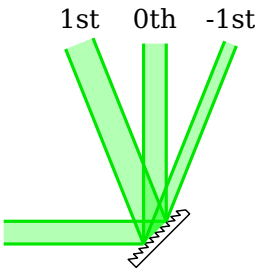
As example we consider a telescope which has zero divergence at the input and output. Usually we would need to tweak the refractive index and the lens parameters to get an output beam with no divergence. Instead, we draw the three beam parts separately using the end points of the previous beams as new starting points and specifying the divergence for each beam part explicitely.



```
\begin{pspicture}(4,2)
  \pnode(0,1){A}\pnode(4,1){B}
  \begin{optexp}
    \lens[lens=0.5 0.5 0.5, abspos=0.5](A)(B)
    \lens[lens=4 4 2, abspos=2](A)(B)
    \addtopsstyle{Beam}{%
      fillstyle=solid, fillcolor=green, opacity=0.3}
    \psset{loadbeampoints}
    \drawwidebeam[beamwidth=0.2, stopinside](A){1}
    \drawwidebeam[beamdiv=-60]{1}{2}
    \drawwidebeam{2}(B)
  \end{optexp}
\end{pspicture}
```

The next example shows diffraction from a grating in the zeroth and first diffraction order. First we draw the beam to the grating, its end points are used for the beams into both diffraction orders.

Ex. 8.1




```
\begin{pspicture}(3.5,3.5)
  \pnode(0,0.5){A}\pnode(2,0.5){B}\pnode(2,3){C}
  \nodexn{(C)-(1,0)}{C'}
  \nodexn{(C)(1,0)}{C''}
  \optgrating(A)(B)(C)
  \addtopsstyle{Beam}{%
    fillstyle=solid, fillcolor=green, opacity=0.3}
  \drawwidebeam[beamwidth=0.3](A){1}
  \psset{savebeampoints=false, loadbeampoints}
  \drawwidebeam{1}(C)
  \drawwidebeam{1}(C')
  \drawwidebeam{1}(C'')
\end{pspicture}
```

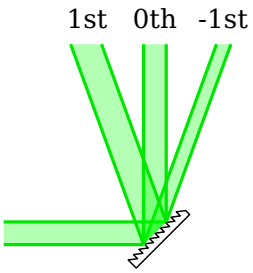
savebeam=true, false default: true

Saves the end conditions, i.e. end points and output vectors, of each \draw*beam macro, previous values are overwritten. \drawbeam and \drawwidebeam are treated separately. If a number $\langle int \rangle > 0$ is given, you can save as many end points as you like, but in most use cases enabling and disabling this option is sufficient.

loadbeam=true, false default: false

Use the end condition (points and vectors) of a previous beam as starting conditions for the next beam. The parameter beamangle can be used to change the direction relative to the loaded one (see following example).

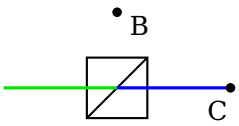
Ex. 8.2



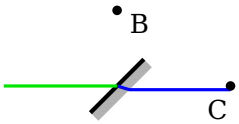

```
\begin{pspicture}(3.5,3.5)
  \pnode(0,0.5){A}\pnode(2,0.5){B}\pnode(2,3){C}
  \optgrating(A)(B)(C)
  \addtopsstyle{Beam}{%
    fillstyle=solid, fillcolor=green, opacity=0.3}
  \drawwidebeam[savebeam, beamwidth=0.3](A){1}
  \psset{savebeam=false, loadbeam}
  \drawwidebeam[beamangle=-20]{1}(C)
  \drawwidebeam{1}(C)
  \drawwidebeam[beamangle=20]{1}(C)
\end{pspicture}
```

Please note, that loadbeam and savebeam do not always work, if the beam path ends inside a semitransparent mirror or a beamsplitter. In these cases only the transmitted ray vector is saved and continuing the beam via the reflective path is not possible.

Accordingly, in the following examples the red beam is not drawn, which should continue the green beam from the component to node ⟨B⟩.




```
\begin{pspicture}(3,1.5)
  \pnode(0,0.5){A}\pnode(1.5,1.5){B}\pnode(3,0.5){C}
  \beamsplitter(A)(1.5,0.5)(B)
  \drawbeam[beaminsidelast, savebeam](A){}
  \drawbeam[linecolor=red, beaminsidefirst, loadbeam]{}(B)
  \drawbeam[linecolor=blue, beaminsidefirst, loadbeam]{}(C)
\end{pspicture}
```



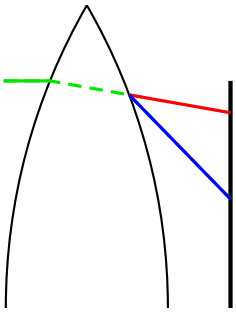

```
\begin{pspicture}(3,1.5)
  \pnode(0,0.5){A}\pnode(1.5,1.5){B}\pnode(3,0.5){C}
  \mirror[mirrortype=semitrans](A)(1.5,0.5)(B)
  \drawbeam[beaminsidelast, savebeam](A){}
  \drawbeam[linecolor=red, beaminsidefirst, loadbeam]{}(B)
  \drawbeam[linecolor=blue, beaminsidefirst, loadbeam]{}(C)
\end{pspicture}
```

startinside=true, false default: false

The parameter beaminsidefirst controls if the internal beams are drawn inside the first component. However, for custom beams it can also be necessary to trace the internal beam of the first component without drawing it, e.g. if the previous beam ended at the first interface.

In the following example the red beam is the wrong continuation of the incoming green beam because it assumes that the loaded beam conditions refer to the output interface. But the previous beam stopped at the input interface, so

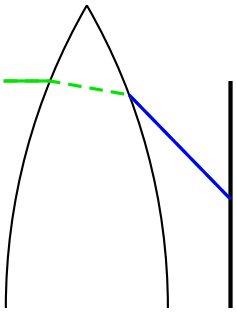
one needs to set startinside in order to continue the beam properly (see blue beam). The dashed line helps to understand the correct beam path through the lens.



```
\begin{pspicture*}(0,2)(3.05,6)
  \optplane(0,2)
  \lens[lens=8 8 8, abspos=0.6, n=1.8](0.5,2)(3,2)
  \optplate[position=1, plateheight=6](0.5,2)(3,2)
  \psset{savebeam=false}
  \drawbeam[beampos=3, linestyle=dashed]{-}
  \drawbeam[beampos=3, savebeam]{1-2}
  \drawbeam[loadbeam, linecolor=red]{2-3}
  \drawbeam[loadbeam, startinside, linecolor=blue]{2-3}
\end{pspicture*}
```

stopinside=true, false default: false

This option is equivalent to startinside but refers to the last component.



```
\begin{pspicture*}(0,2)(3.05,6)
  \optplane(0,2)
  \lens[lens=8 8 8, abspos=0.6, n=1.8](0.5,2)(3,2)
  \optplate[position=1, plateheight=6](0.5,2)(3,2)
  \psset{savebeam=false}
  \drawbeam[beampos=3, linestyle=dashed]{-}
  \drawbeam[beampos=3, savebeam, stopinside]{1-2}
  \drawbeam[loadbeam, linecolor=blue]{2-3}
\end{pspicture*}
```

8.6. Drawing fibers

All available components can be connected with fiber connections, the fiber components are connected automatically.

\drawfiber[⟨*options*⟩]{⟨obj_1⟩}{⟨obj_2⟩}...

The \drawfiber command takes ID, compname, or nodes as ⟨obj_N⟩ arguments (see Sec. 8.1), but ID ranges like with \drawbeam are not possible. If you specify more than two arguments, you get connections $1 \to 2$, $2 \to 3$, and so on.

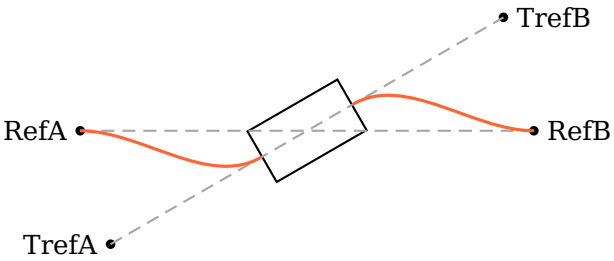
The automatic fiber connections are drawn internally with \drawfiber commands from each reference node to the component.

8.6.1. Fiber angles

The fibers are usually drawn as \nccurve connections, the angles at both curve ends are automatically determined by the component orientation and depend on whether you connect a component with a component, or a component with a node.

Component with node

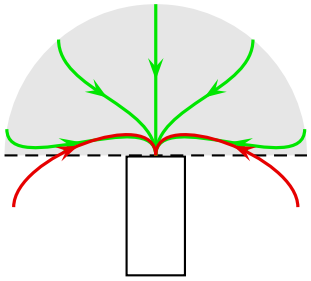
A connection of a component with a node is what is used for automatic connections (see Sec. 8.6.3). Consider the following example for clarification of the curve alignment:




```
\begin{pspicture}(-1,-0.7)(7,2.7)
  \pnode(0,1){A}\pnode(6,1){B}
  \optbox[angle=30](A)(B)
  \drawfiber(A){}(B)
\end{pspicture}
```

The angle at the component itself agrees with the transformed reference line. The angle at the node corresponds to the angle of the original reference line of the component which it gets connected to. The fiber is always directed to the component node.

Considering the following example, this means that the start angles of the red fibers are rotated by 180° with respect to the green fibers. The angles at the component do not change.

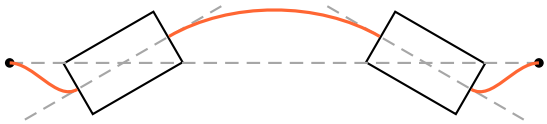


```
\begin{pspicture}(-2,-1.6)(2,2)
  \optbox[position=end](0,1)(0,0)
  \addtopsstyle{Fiber}{linecolor=green!90!black}
  \multido{\i=10+40}{5}{%
    \drawfiber(2;\i){}
  }
  \addtopsstyle{Fiber}{linecolor=red!90!black}
  \drawfiber(2;-20){}
  \drawfiber(2;200){}
\end{pspicture}
```

Component with component

This variant is very helpful to achieve smoother connections between components which are not aligned on the same reference line but are arranged maybe in a loop or similar, see Ex. 4 and Ex. 8.6.2.

In this case, the curve angles are determined by the component which the respective curve end is connected to. The fibers are always directed away from the component center.



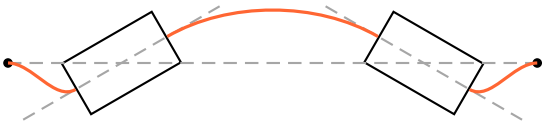
```
\begin{pspicture}(0,-0.8)(7,0.8)
  \pnode(0,0){A}\pnode(3,0){B}
  \pnode(4,0){C}\pnode(7,0){D}
  \optbox[angle=30](A)(B)
  \optbox[angle=-30](C)(D)
  \drawfiber(A){1}{2}(D)
\end{pspicture}
```

These automatically calculated start and stop angles can be changed in a variety of ways, the available parameters are listed in the following.

`fiberalign`=`rel`, `relative`, `center`, `abs`, `absolute` default: `relative`

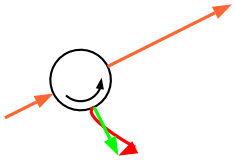
Set the reference for the fiber angles.

`relative` The fiber angles are relative to the component. When connecting a node to a component, the angle at the node is that of the component reference line. The angle at the component agrees with that of its transformed reference line.



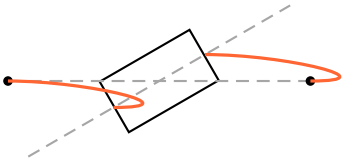
```
\begin{pspicture}(1,-1.7)(7,1.7)
  \pnode(0,0){A}\pnode(3,0){B}
  \pnode(4,0){C}\pnode(7,0){D}
  \optbox[angle=30](A)(B)
  \optbox[angle=-30](C)(D)
  \drawfiber(A){1}{2}(D)
\end{pspicture}
```

`center` The fiber angles both at the node and the component are relative to the connection between the involved interface node and the center node of the component. In most cases this is equivalent to `relative`, but necessary e.g. for `\optcirculator`. In the example, the green fiber shows the correct alignment, while the red one is with `relative` alignment.



```
\begin{pspicture}(1,0)(3,2)
  \addtopsstyle{Fiber}{arrowscale=1.3, arrows=->,
      arrowinset=0}
  \optcirculator[fiber=t](0,0.5)(3,2)(1.5,0)
  \drawfiber[linecolor=red]{}(1.5,0)
  \drawfiber[fiberalign=center, linecolor=green]{}(1.5,0)
\end{pspicture}
```

`absolute` The fiber angles are absolute, no automatic values are added.



```
\begin{pspicture}(1,-1)(4.5,1)
  \pnode(0,0){A}\pnode(4,0){B}
  \optbox[angle=30](A)(B)
  \drawfiber[fiberalign=abs, ncurv=1.5](A){}(B)
\end{pspicture}
```

`fiberangleA`=⟨*num*⟩ default: `0`

An additive value for the starting angle, the total angle depends on `fiberalign`. If this parameter is set to `absolute`, then `fiberangleA` is the total angle, otherwise its value is added to the calculated angle.

`fiberangleB`=⟨*num*⟩ default: `0`

Like `fiberangleA`, but for the end angle.

`startnode`=`auto`, `N`, `1`, `2`, `...` default: `auto`

The connections with `\drawfiber` are drawn between the two nodes which are nearest to each other. This fits most of the time, but not always. With this option you can choose the start node explicitely.

`stopnode`=`auto`, `N`, `1`, `2`, `...` default: `auto`

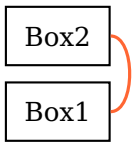
Like `startnode`, but for the stop node.

The following examples show some use cases for `startnode` and `stopnode`.

1. This example shows a situation, where choosing the nearest nodes is correct.

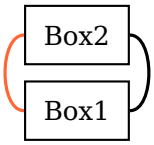

```
\begin{pspicture}(3,2)
  \pnode(-0.5,1){A}\pnode(1.5,1){B}\pnode(3.5,1){C}
  \psset{label=0, optboxwidth=1}
  \optbox(A)(B){Box1}\optbox(B)(C){Box2}
  \drawfiber{1}{2}
\end{pspicture}
```

2. In this case, the nearest connection is not well defined because two connections have the same length.




```
\begin{pspicture}(3,2)
  \pnode(0,0.5){A}\pnode(3,0.5){B}
  \pnode([offset=1]B){C}\pnode(A|C){D}
  \psset{label=0}
  \optbox(A)(B){Box1}\optbox(C)(D){Box2}
  \drawfiber{1}{2}
\end{pspicture}
```

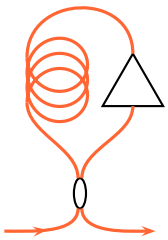
3. To select the desired connection it is often enough to set either `startnode` or `stopnode`. With this you fix one node, the other is the one nearest to it. In rare cases it might be necessary to fix both nodes.




```
\begin{pspicture}(2,2)
  \pnode(0,0.5){A}\pnode(2,0.5){B}
  \pnode([offset=1]B){C}\pnode(A|C){D}
  \psset{label=0}
  \optbox(A)(B){Box1}\optbox(C)(D){Box2}
  \drawfiber[startnode=1]{1}{2}
  \drawfiber[startnode=N, linecolor=black]{1}{2}
\end{pspicture}
```

4. Another example where you need to specify the `startnode`.

Ex. 8.3



```
\begin{pspicture}(2.1,3)
  \pnode(0,0){A}\pnode(2,0){B}
  \pnode(0.3,3){C}\pnode(1.7,3){D}
  \psset{fiber=none}
  \optcoupler[fiber=l,
              addtoFiberIn1={angleA=0, ArrowInside=->},
              addtoFiberIn2={angleA=180, arrows=<-},
              abspos=0.5, compname=Cpl](A)(B)(C)(D)
  \optfiber[compname=Hnlf, abspos=1](C)(C|A)
  \optamp[abspos=2, compname=Amp](D|B)(D)
  \drawfiber{Cpl}{Hnlf}
  \drawfiber[startnode=1, ncurv=1.2]{Hnlf}{Amp}
  \drawfiber{Cpl}{Amp}
\end{pspicture}
```

8.6.2. Fiber appearance

`Fiber`⟨*psstyle*⟩

The appearance of fibers is controlled with this style. The optional argument of `\drawfiber` can be used as well, these options can overwrite settings from the `Fiber` style.

`addtoFiber`=⟨*list*⟩

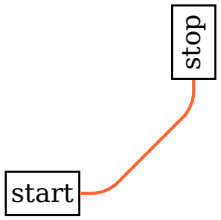
The `Fiber` style is extended by locally adding the parameters contained in ⟨*list*⟩. The ⟨*list*⟩ must be surrounded by curly braces.

`newFiber`=⟨*list*⟩

Similar to `addtoFiber`, but an existing `Fiber` style is overwritten with the new parameter set.

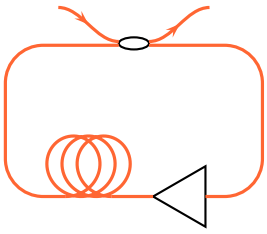
`fiberstyle`=⟨*string*⟩ default: `curve`

The fibers are usually drawn as `\nccurve`, but any type of node connection `\nc`⟨*string*⟩ from `pst-node` can be selected with this parameter, see the `pst-node` documentation[3] for possible values. The following examples show possible use cases, Ex. 10 contains another typical use case.




```
\begin{pspicture}(3,3)
  \pnode(1, 0.5){A}\pnode(2.5,0.5){B}\pnode(2.5,2){C}
  \psset{optboxsize=1 0.6, innerlabel}
  \optbox[position=start](A)(B){start}
  \optbox[position=end](B)(C){stop}
  \drawfiber[fiberstyle=diag, linearc=0.5]{1}{2}
\end{pspicture}
```

Ex. 8.4



```
\begin{pspicture}(4,2.5)
  \pnode(1,2.5){In}\pnode(3,2.5){Out}
  \pnode(0,2){A}\pnode(4,2){B}\pnode(4,0){C}\pnode(0,0){D}
  \optcoupler[fiber=t, addtoFiber={ArrowInside=->}, )
      coupleralign=b](In)(A)(Out)(B)
  \psset{fiber=none}
  \optamp[position=0.35](C)(D)
  \optfiber[position=0.35](D)(C)
  \drawfiber{2}{3}
  \addtopsstyle{Fiber}{fiberstyle=bar, linearc=0.5, armA=1.5}
  \drawfiber[stopnode=1]{1}{2}
  \drawfiber[stopnode=1]{1}{3}
\end{pspicture}
```

[3] http://mirror.ctan.org/help/Catalogue/entries/pst-node.html

8.6.3. Automatic fiber connections

All fiber components described in Sec. 5 are connected automatically to their reference nodes with \drawfiber, their behaviour is descibed in Sec. 8.6.1. The fiber connections can be controlled in a number of ways, without need to explicitely use \drawfiber. The style of each fiber connection can be configure independently, see Sec. 8.6.4.

fiber=[*+]none, all, i, o, ⟨*refpoint*⟩ default: all

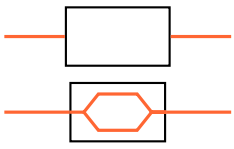
This parameter selects which fiber connections are drawn. The value of this parameter can affect either only the fiber components (if the value is prefixed with *), only free-ray components (prefixed with +), or all components otherwise. Typical free-ray components which are often used with fibers as well are \optbox and \optdetector.

You can select to connect all nodes, none, or select distinct connections to draw. Value i (l is equivalent) draws all input fibers, o (r is equivalent) draws all output fibers. These values are exclusive, that means io does not select all fibers, but none.

Components which have more than one input or output fiber (couplers and \optcirculator) allow also values t and b, possibly in combination with i, o, r, or l.

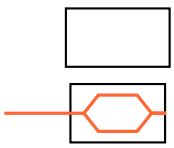
See the following examples for some variants.

1. Enable all fiber connections for all components.



```
\begin{pspicture}(3,2)
  \psset{fiber}
  \optbox(0,1.5)(3,1.5)
  \optmzm(0,0.5)(3,0.5)
\end{pspicture}
```

2. Draw only the input fiber of a fiber component (\optmzm), the free-ray component (\optbox) is not affected.



```
\begin{pspicture}(3,2)
  \psset{fiber=*i}
  \optbox(0,1.5)(3,1.5)
  \optmzm(0,0.5)(3,0.5)
\end{pspicture}
```

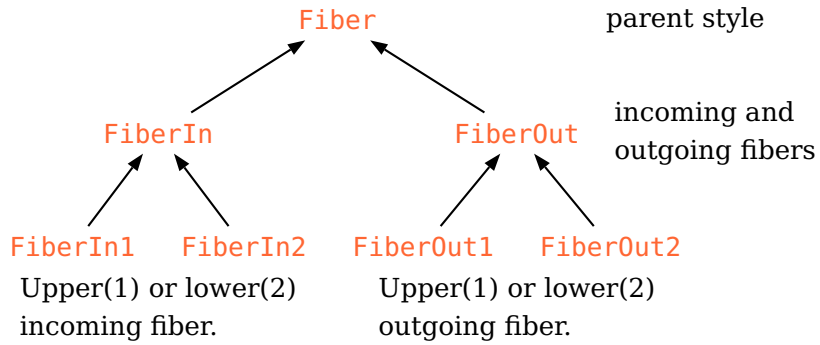
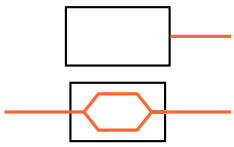


Figure 8.1.: Inheritance diagram for the psstyles used for the automatic fiber connections. These styles should be changed with `\addtopsstyle` or the respective `addto`⟨*style*⟩ options to preserve the inheritance.

3. Draw only the output fiber of a free-ray component (`\optbox`), but all connections of the fiber component (`\optmzm`).



```
\begin{pspicture}(3,2)
  \psset{fiber=+o}
  \optbox(0,1.5)(3,1.5)
  \optmzm(0,0.5)(3,0.5)
\end{pspicture}
```

4. The values `t` and `b` affect only components which have more than two fibers, i.e `\optcirculator` and the couplers (Sec. 5.10).


```
\begin{pspicture}(3,1)
  \optcoupler[fiber=t](0,1)(0,0)(3,1)(3,0)
\end{pspicture}
```

```
\begin{pspicture}(3,1)
  \wdmsplitter[fiber=br](0,0.5)(3,1)(3,0)
\end{pspicture}
```

```
\begin{pspicture}(3,1)
  \optcirculator[fiber=b](0,0.5)(3,1)(1,0)
\end{pspicture}
```

8.6.4. Appearance of automatic fiber connections

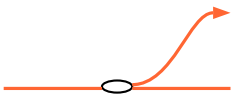
The styles of all automatic fiber connections can be configured independently, they are all derived from the basic `Fiber` style which is described in Sec. 8.6.2.

FiberIn
FiberIn1
FiberIn2
FiberOut
FiberOut1
FiberOut2

Style FiberIn applies to all input fibers, it inherits from Fiber. If the component has more than one input fiber they have the styles FiberIn1 and FiberIn2 which inherit from FiberIn. All output fibers have the style FiberOut, two output fibers have the styles FiberOut1 and FiberOut2 which inherit from FiberOut. See Fig. 8.1 for an inheritance diagram of the styles. You must have in mind, that this inheritance is effective only if you use \addtopsstyle to change the styles (see also below).

new⟨*style*⟩
addto⟨*style*⟩

For every style two appropriate keys new⟨*style*⟩ and addto⟨*style*⟩ are provided which can be used to change the styles for single objects. This can be used to define own components (Sec. 9.1) with respectively changed fiber connections or to avoid explicit grouping.



99/1

```
\begin{pspicture}(3,2)
  \newpsobject{tapcoupler}{wdmsplitter}{%
    coupleralign=b,
    addtoFiberOut1={arrows=->, arrowscale=1.2, arrowinset=0}}
  \tapcoupler(0,0.5)(3,1.5)(3,0.5){99/1}
\end{pspicture}
```

8.7. Layers

The necessary order of constructing a drawing is to define the components first before you can connect them. This implies that all connections are drawn on top of the components, which might be unwanted.

To circumvent this, we provide the optexp environment, inside which all connections are drawn behind the components.

```
\begin{optexp}
  \end{optexp}
```

no layers

with layers

```
\begin{pspicture}(3,4)
  \psset{beam, fillstyle=solid, fillcolor=black}
  \optbox(0,3.5)(3,3.5){no layers}
  \begin{optexp}
    \optbox(0,1.5)(3,1.5){with layers}
  \end{optexp}
\end{pspicture}
```

Beware, that this is achieved by executing twice all code included in the environment. This works fine for all `pst-optexp` commands which take care of not drawing stuff twice but only in a changed order. All other text and graphics are drawn twice which leads to ugly results because of altered anti-aliasing in the viewer.

To avoid this, you can either move all other code outside the environment, or use the following two commands to define code parts which are executed only once.

`\backlayer`{⟨*code*⟩}

Execute ⟨*code*⟩ only once in the first pass.

`\frontlayer`{⟨*code*⟩}

Execute ⟨*code*⟩ only once in the second pass.

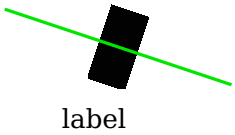
In this example, the text is put on top of the frame although the text is defined first:


```
\begin{pspicture}(3,1)
\begin{optexp}
  \frontlayer{\rput(1.5,0.5){front}}
  \backlayer{\psframe*[linecolor=DOrange!50](1,0)(2,1)}
\end{optexp}
\end{pspicture}
```

9. Custom components

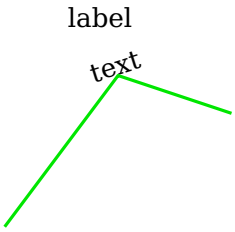
The `pst-optexp` package provides two commands which can use anything as optical components. This includes e.g. external images or your own drawings.

`\optdipole`[⟨*options*⟩](⟨*in*⟩)(⟨*out*⟩){⟨*label*⟩}




```
\begin{pspicture}(3,3)
  \optdipole[labeloffset=1, beam](0,2)(3,1){%
    \rput(0,0){\rule{0.5cm}{1cm}}%
  }{label}
\end{pspicture}
```

`\opttripole`[⟨*options*⟩](⟨*in*⟩)(⟨*center*⟩)(⟨*out*⟩){⟨*label*⟩}

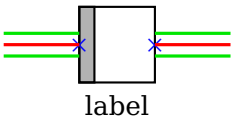



```
\begin{pspicture}(3,3)
  \opttripole[beam](3,1.5)(1.5,2)(0,0){%
    \rput[b](0,0){text}%
  }{label}
\end{pspicture}
```

These components can also be connected with beams and fibers, but do not provide full access to all options. `\opttripole` defines a single reflective interface, whereas `\optdipole` can have two transmittive interfaces:

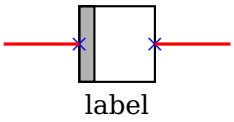
`optdipolesize`=⟨*width*⟩[⟨*height*⟩] default: `0 0`

With this option you can define two transmittive interfaces for `\optdipole`, which are placed at $(\pm 0.5\langle width\rangle, 0)$.




```
\begin{pspicture}(3,2)
\pnode(0,1){A}\pnode(3,1){B}
\optdipole[optdipolesize=1, showifcnodes](A)(B){%
  \psframe[fillstyle=solid,fillcolor=black!30](-0.5,-0.5)(-0.3,0.5)
  \psframe(-0.5,-0.5)(0.5,0.5)}{label}
\drawwidebeam[beamwidth=0.3, beaminside=false](A){}(B)
\drawbeam[beaminside=false, linecolor=red](A){}(B)
\end{pspicture}
```

If also a ⟨*height*⟩ greater than zero is set, this is used as size of the numerical aperture for ray tracing (see useNA). In the following example the height is set to `0.2` which is smaller than the wide beam which is not drawn, in contrast to the previous example.

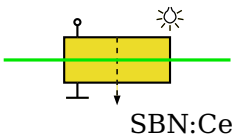



```
\begin{pspicture}(3,2)
\pnode(0,1){A}\pnode(3,1){B}
\optdipole[optdipolesize=1 0.2, showifcnodes](A)(B){%
  \psframe[fillstyle=solid,fillcolor=black!30](-0.5,-0.5)(-0.3,0.5)
  \psframe(-0.5,-0.5)(0.5,0.5)}{label}
\drawwidebeam[beamwidth=0.3, beaminside=false](A){}(B)
\drawbeam[beaminside=false, linecolor=red](A){}(B)
\end{pspicture}
```

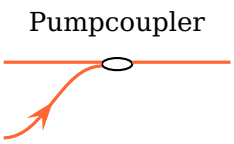
Further examples are Ex. 10 and Ex. 10.

9.1. Customized versions of existing components

You can define a customized version of an existing component using the `\newpsobject` macro. With this you can define a new component using predefined objects with a set of options. These options serve only as default values and can be overridden when calling the macro.

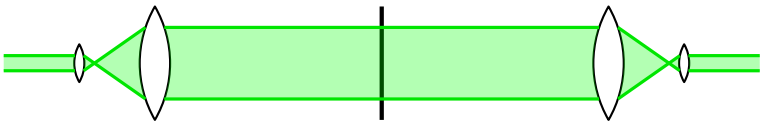



```
\begin{pspicture}(3,1.7)
\newpsobject{sbn}{crystal}{%
  voltage, lamp, %
  fillstyle=solid, fillcolor=yellow!90!black}
\sbn[label=1.2 45, beam](0,1)(3,1){SBN:Ce}
\end{pspicture}
```




```
\begin{pspicture}(3,1.7)
\newpsobject{pumpcoupler}{wdmcoupler}{%
  coupleralign=top, addtoFiberIn2={ArrowInside=->, arrowscale=2}}
\pumpcoupler[label=0.5 180](0,1)(0,0)(3,1){Pumpcoupler}
\end{pspicture}
```

Ex. 9.1



```
\begin{pspicture}(10,2)
  \newpsobject{MOLensIn}{lens}{lens=0.5 0.5 0.5}
  \newpsobject{MOLensOut}{lens}{lens=1.5 1.5 1.5}
  \pnode(0,1){A}\pnode(10,1){B}
  \MOLensIn[abspos=1](A)(B)\MOLensOut[abspos=2](A)(B)
  \optplate[plateheight=1.5](A)(B)
  \MOLensOut[abspos=8](A)(B)\MOLensIn[abspos=9](A)(B)
  \addtopsstyle{Beam}{n=1, fillstyle=solid, fillcolor=green, opacity=0.3}
  \psset{loadbeampoints}
  \drawwidebeam[beamwidth=0.2, stopinside](A){1}
  \drawwidebeam[beamdiv=-70]{1}{2}
  \drawwidebeam[stopinside]{2-4}
  \drawwidebeam[beamdiv=-70]{4}{5}
  \drawwidebeam{5}(B)
\end{pspicture}
```

9.2. Defining new objects

`pst-optexp` provides some high-level macros to allow very convenient definition of your own custom components.

Note, that this part of the documentation is not complete. It roughly describes the steps to define own components as of version 3.0. Some parts may be subject to change which are not backward compatible.

The old syntax of defining `\mycomp@iii` is supported for backward compatibility, but will be dropped soon. Please adapt your macros to the new syntax.

`\newOptexpDipole`[⟨*fixopt*⟩]{⟨*name*⟩}{⟨*dftopt*⟩}

`\newOptexpTripole`[⟨*fixopt*⟩]{⟨*name*⟩}{⟨*dftopt*⟩}

`\newOptexpFiberDipole`[⟨*fixopt*⟩]{⟨*name*⟩}{⟨*dftopt*⟩}

These macros generate all organizing code for the components which handle the positioning, label placement, rotation and shifting, and the layering. In the simplest case you only have to define the actual drawing of the component outline.

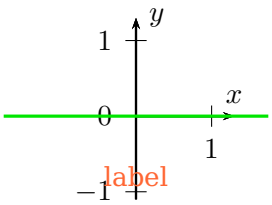
The macro `\newOptexpDipoleNolabel` is superfluous because since version 3.0 the label is optional for all components.

The actual process of creating a new component is split up in different parts, depending on the component complexity and user requirements:

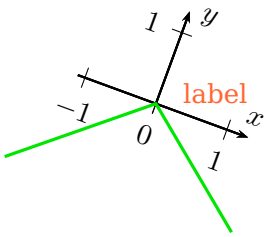
1. The component drawing (see Sec. 9.2.1), this is the only required part.
2. Support for raytracing and layers (optional)
3. Optional support for `rotateref`, `extnode` and `position=start|end`.

9.2.1. The component drawing

You must define a macro which is called e.g. `\...@comp`, which contains the component drawings. See the following two examples for dipole and tripole which show the orientation and position of the component coordinate system.



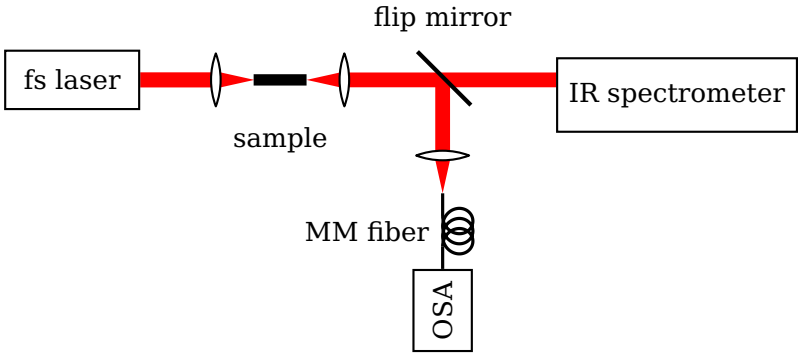

```
\begin{pspicture}(3.5,2.6)
\newOptexpDipole{mydipole}{}
\makeatletter
\def\mydipole@comp{%
  \psaxes[arrows=->](0,0)(0,-1.1)(1.3,1.3)[$x$,90][$y$,0]
}%
\makeatother
\mydipole(0,1)(3.5,1){\color{spot}label}
\drawbeam(0,1){}(3.5,1)
\end{pspicture}
```




```
\begin{pspicture}(3.5,3.7)
\newOptexpTripole{mytripole}{labelangle=-60}
\makeatletter
\def\mytripole@comp{%
  \psaxes[arrows=->](0,0)(-1.1,0)(1.3,1.3)[$x$,90][$y$,0]
}%
\makeatother
\mytripole(0,1)(2,1.7)(3,0){\color{spot}label}
\drawbeam(0,1){}(3,0)
\end{pspicture}
```

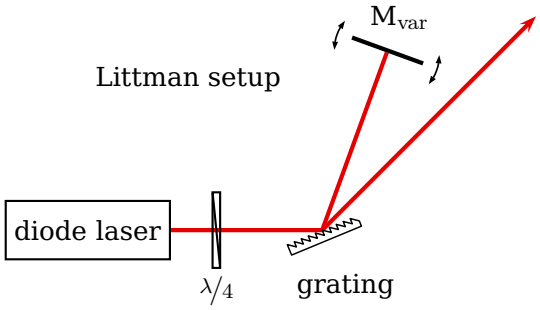
10. Examples

Ex. 10.1



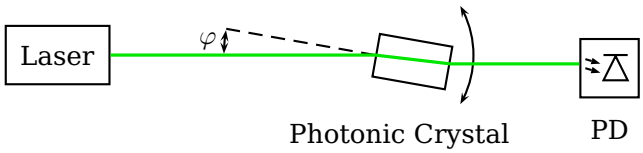

```
\begin{pspicture}(0.1,0.4)(10,4.5)
  \psset[optexp]{lens=1 1 0.7, loadbeampoints}
  \pnode(2,4){L}\pnode([Xnodesep=4]L){M}
  \pnode([Xnodesep=1.5]M){IRSpec}\pnode([offset=-2.5]M){OSA}
  \begin{optexp}
  \optbox[position=start, innerlabel, optboxwidth=1.8](L)(M){fs laser}
  \lens[abspos=1](L)(M)
  \optbox[abspos=1.85, optboxsize=0.7 0.15, fillstyle=solid,
      fillcolor=black](L)(M){sample}
  \lens[abspos=2.7](L)(M)
  \mirror[labelangle=45](L)(M)(OSA){flip mirror}
  \optbox[position=end, optboxsize=3.2 1, innerlabel,
      compshift=-0.2](M)(IRSpec){IR spectrometer}
  \lens[abspos=1](M)(OSA)
  \optplane[angle=90]([offset=-1.5]M)
  \optfiber[fiberloopradius=0.2, fiberloopsep=0.2, label=1,
      newFiber={linewidth=1.5\pslinewidth}]([offset=-1.5]M)(OSA){MM fiber}
  \optbox[position=end, innerlabel, optboxwidth=1.1](M)(OSA){OSA}
  \newpsstyle{Beam}{linestyle=none, fillcolor=red, fillstyle=solid, raytrace=false}
  \drawwidebeam[beamwidth=0.2, stopinside]{1-2}
  \drawwidebeam[beamdiv=-25]{2-3}
  \drawwidebeam[loadbeampoints=false, beamdiv=25]{3-4}
  \drawwidebeam[startinside]{4-5}
  \drawwidebeam[savebeampoints=false, beamalign=abs]{5-6}
  \drawwidebeam[stopinside]{5}{7}
  \drawwidebeam[beamdiv=-25]{7-8}
\end{optexp}
\end{pspicture}
```

Ex. 10.2



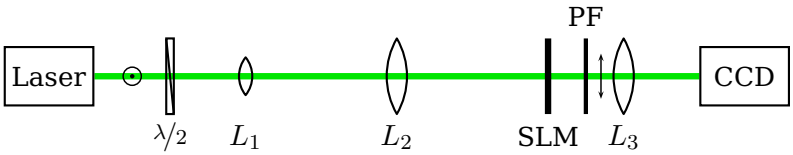

```
\begin{pspicture}(-4.2,-1)(3,3)
\pnode(-2,0){LOut}\pnode(0,0){Grat}\pnode(4;45){Out}\pnode(2.5;70){Mvar}
\newpsstyle{Beam}{linewidth=2\pslinewidth, linecolor=red!90!black}
\begin{optexp}
  \optbox[optboxwidth=2.2, labeloffset=0, position=start](LOut)(Grat){diode laser}
  \mirror[variable, label=0.5](Grat)(Mvar)(Grat){M$_\mathrm{var}$}
  \optretplate[position=0.3](LOut)(Grat){$\nicefrac{\lambda}{4}$}
  \optgrating(LOut)(Grat)(Out){grating}
  \drawbeam[arrows=->]{1}{3}{4}(Out)
  \drawbeam{2}{4}
\end{optexp}
\rput[l](-3,2){Littman setup}
\end{pspicture}
```

Ex. 10.3



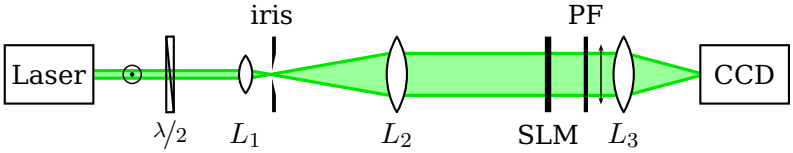

```
\begin{pspicture}(8.5,1.9)
\pnode(1.4,1.3){Laser}\pnode(7.6,1.3){Diode}
\optbox[position=start, labeloffset=0](Laser)(Diode){Laser}%
\optbox[abspos=4, optboxsize=1 0.6, labeloffset=1, n=3,
       compname=PC, angle=-10, rotateref=l](Laser)(Diode){Photonic Crystal}
\optdetector[dettype=diode](\oenodeOut{PC})(Diode|\oenodeOut{PC}){PD}
\nodexn{(\oenodeIn{PC}) + (2;170)}{Angle1}
\psline[linestyle=dashed](\oenodeIn{PC})(Angle1)
\psarc{<->}(\oenodeIn{PC}){1.3}{330}{30}
\psarc[arcsep=1pt]{<->}(\oenodeIn{PC}){2}{170}{180}
\uput{2.1}[175](\oenodeIn{PC}){\small $\varphi$}
\drawbeam{-}
\end{pspicture}
```

Ex. 10.4



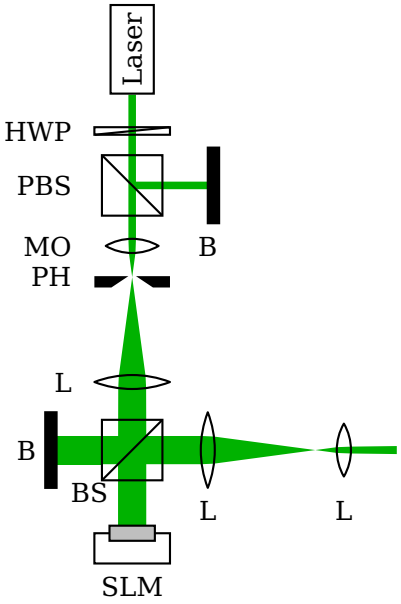

```
\begin{pspicture}(10.4,1.7)
\pnode(1.2,1){Start}\pnode(9.2,1){CCD}
\begin{optexp}
\optbox[position=end, label=0, optboxwidth=1.2](CCD)(Start){Laser}
\optbox[position=end, label=0, optboxwidth=1.2](Start)(CCD){CCD}
\polarization[poltype=perp,abspos=0.5](Start)(CCD)
\optretplate[abspos=1](Start)(CCD){$\nicefrac{\lambda}{2}$}
\lens[lens=0.4 0.4 0.5,abspos=2](Start)(CCD){$L_1$}
\lens[abspos=4](Start)(CCD){$L_2$}
\optplate[abspos=6, platelinewidth=3\pslinewidth](Start)(CCD){SLM}
\optplate[abspos=6.5, labelangle=180](Start)(CCD){PF}
\polarization[abspos=6.7](Start)(CCD)
\lens[abspos=7](Start)(CCD){$L_3$}
\drawbeam[linewidth=2\pslinewidth]{1}{2}
\end{optexp}
\end{pspicture}
```

Ex. 10.5



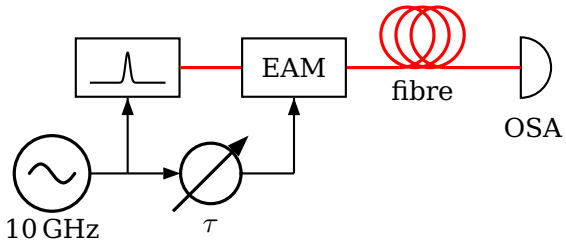

```
\begin{pspicture}(10.4,1.7)
\pnode(1.2,1){Start}\pnode(9.2,1){CCD}
\begin{optexp}
\optbox[position=start, label=0, optboxwidth=1.2](Start)(CCD){Laser}
\polarization[poltype=perp,abspos=0.5](Start)(CCD)
\optretplate[abspos=1](Start)(CCD){$\nicefrac{\lambda}{2}$}
\lens[lens=0.4 0.4 0.5,abspos=2](Start)(CCD){$L_1$}
\pinhole[phwidth=0.05, abspos=2.35, labelangle=180](Start)(CCD){iris}
\lens[abspos=4](Start)(CCD){$L_2$}
\optplate[abspos=6,platelinewidth=3\pslinewidth](Start)(CCD){SLM}
\optplate[abspos=6.5,labelangle=180](Start)(CCD){PF}
\lens[abspos=7](Start)(CCD){$L_3$}
\optbox[position=end, label=0, optboxwidth=1.2](Start)(CCD){CCD}
\polarization[abspos=6.7, polsize=0.8](Start)(CCD)
\addtopsstyle{Beam}{fillstyle=solid, fillcolor=green!40!white, loadbeampoints}
\drawwidebeam[beamwidth=0.1, stopinside]{1}{4}
\drawwidebeam[beamdiv=-20]{4}{6}
\drawwidebeam[stopinside]{6-9}
\drawwidebeam[beamdiv=-30]{9-10}
\end{optexp}
\end{pspicture}
```

Ex. 10.6: Adapted from arXiv:1112.5270v2



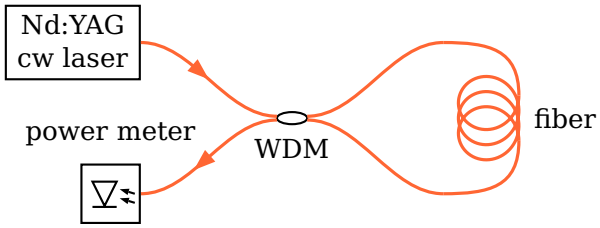

```
\begin{pspicture}(0.3,0.4)(5.5,8.3)
\pnode(2,7){L}\pnode(2,5.8){Pbs}\pnode([Xnodesep=1]Pbs){Blk1}\pnode(2,2.3){Bs}
\pnode([Xnodesep=-1]Bs){Blk2}\pnode([Xnodesep=3.5]Bs){Out}\pnode(2,1.3){Slm}
\begin{optexp}
  \optbox[position=end, innerlabel, optboxsize=1.2 0.6](Pbs)(L){Laser}
  \psset{labelalign=r, mirrortype=extended}
  \newpsstyle{ExtendedMirror}{fillstyle=solid, fillcolor=black}
  \optretplate[position=0.4](L)(Pbs){HWP}
  \beamsplitter[labelangle=-90](L)(Pbs)(Blk1){PBS}
  \mirror[label=. -90 c](Pbs)(Blk1)(Pbs){B}
  \lens[abspos=0.8, lens=0.7 0.7 0.7, n=2](Pbs)(Bs){MO}
  \pinhole[phwidth=0.15, abspos=1.2](Pbs)(Bs){PH}
  \lens[abspos=2.6, lensradius=1.4](Pbs)(Bs){L}
  \psset{labelalign=c}
  \beamsplitter[labelangle=-135](Pbs)(Bs)(Blk2){BS}
  \mirror[labeloffset=0.4](Bs)(Blk2)(Bs){B}
  \lens[abspos=1, lensradius=1.4](Bs)(Out){L}
  \lens[abspos=2.8, lens=0.7 0.7 0.7, n=2](Bs)(Out){L}
  \opttripole(Bs)(Slm)(Bs){%
    \psframe[fillstyle=solid,fillcolor=gray!50](-0.3,0)(0.3,0.2)
    \psline(-0.3,0.1)(-0.5,0.1)(-0.5,0.5)(0.5,0.5)(0.5,0.1)(0.3,0.1)}{SLM}
  \addtopsstyle{Beam}{linestyle=none, fillstyle=solid, fillcolor=green!70!black}
  \drawwidebeam[beamwidth=0.1]{1-4}
  \drawwidebeam[beamwidth=0.1, beaminsidefirst, beaminsidelast]{3}{5-8}
  \psset{savebeampoints=false, loadbeampoints}
  \drawwidebeam{8-11}(Out)\drawwidebeam[beaminsidefirst]{8}{12}
\end{optexp}
\end{pspicture}
```

Ex. 10.7



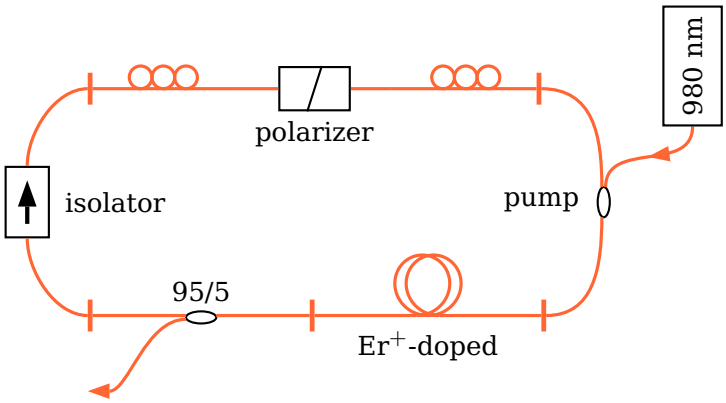

```
\begin{pspicture}(6.4,3.2)
\addtopsstyle{Fiber}{linecolor=red}\psset{fiber=none}
\pnode(2.3,2.3){Lin}\pnode([Xnodesep=4.5]Lin){Det}
\optbox[label=0.2, position=start, compname=L, extnode=b](Lin)(Det){%
  \psGauss[yunit=0.03,sigma=0.03]{-0.5}{0.5}}
\optbox[label=0, compname=EAM, extnode=b, abspos=1.5](Lin)(Det){EAM}
\optfiber[labeloffset=0.3, abspos=3.2](Lin)(Det){fibre}
\optdetector(Lin)(Det){OSA}
\pnode([Xnodesep=-1,offset=-1]\oenodeExt{L}){Osc}
\pnode(\oenodeExt{L}|Osc){PSin}\pnode(\oenodeExt{EAM}|Osc){PSout}
\oscillator[output=right](Osc){10\,GHz}{}
\phaseshifter[arrowscale=1.5, inputarrow, labeloffset=-0.7](PSin)(PSout){$\tau$}
\drawfiber{1}{2}{3}{4}
\psset{arrows=->, arrowinset=0, arrowscale=1.5}
\wire(PSin)(\oenodeExt{L})\wire(PSout)(\oenodeExt{EAM})
\end{pspicture}
```

Ex. 10.8



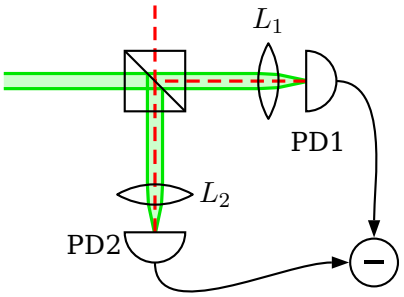

```
\begin{pspicture}(0.2,0.6)(8.2,3.5)
\pnode(2,3){Laser}\pnode(2,1){PwMeter}\pnode(6,3){CplTop}\pnode(6,1){CplBot}
\psset{arrowscale=1.5, arrowinset=0}
\optbox[position=start, optboxsize=1.8 1,
    labeloffset=0](Laser)([Xnodesep=0.1]Laser){%
  \begin{tabular}{@{}c@{}}Nd:YAG\\[-0.4ex]cw laser\end{tabular}}
\optcoupler[addtoFiberIn1={ArrowInside=->}, addtoFiberIn2={ArrowInside=-<},
          labeloffset=0.4](Laser)(PwMeter)(CplTop)(CplBot){WDM}
\optfiber[addtoFiberOut={ncurv=1, angleB=0}, addtoFiberIn={ncurv=1, angleA=0},
        compshift=-1, label=0.2 . l](CplBot)(CplTop){fiber}
\optdetector[dettype=diode]([Xnodesep=0.1]PwMeter)(PwMeter){power meter}
\end{pspicture}
```

Ex. 10.9



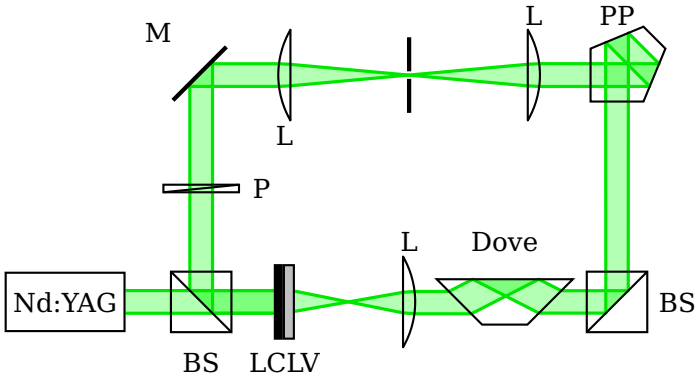

```
\begin{pspicture}(0.9,0.9)(10.4,6.1)
\psset{arrowscale=1.5, arrowinset=0}
\pnode(2,5){PC1in}\pnode(4,5){PC1out}\pnode(6,5){PC2in}
\pnode(8,5){PC2out}\pnode(2,2){CplSig}\pnode(5,2){CplIn}
\pnode(2,1){CplOut}\pnode(10,4.5){Pump}\pnode(8,2){PumpSig}
\optisolator[compshift=0.8, addtoFiberIn={angleA=180},
          addtoFiberOut={angleB=180}, label=0.5 . l]%
          (CplSig)(PC1in){isolator}
\polcontrol[addtoFiberIn={arrows=|-}](PC1in)(PC1out)
\optfiberpolarizer[labeloffset=0.6](PC1out)(PC2in){polarizer}
\polcontrol[addtoFiberOut={arrows=-|}](PC2in)(PC2out)
\wdmsplitter[labeloffset=0.3, coupleralign=bottom, addtoFiberIn={arrows=|-},
          addtoFiberOut1={arrows=->}, addtoFiberOut2={arrows=-|}]%
          (CplIn)(CplOut)(CplSig){95/5}
\wdmcoupler[addtoFiberIn1={ArrowInside=->}, addtoFiberIn2={angleA=0},
         addtoFiberOut={angleB=0,arrows=-|}, ncurv=0.9,
         coupleralign=bottom, compshift=0.8](Pump)(PC2out)(PumpSig){pump}
\optbox[position=start, innerlabel, ↵
   optboxwidth=1.6](Pump)([offset=-0.1]Pump){980~nm}
\optfiber[fiberloops=2, labeloffset=0.4](CplIn)(PumpSig){Er$^+$-doped}
\end{pspicture}
```

Ex. 10.10



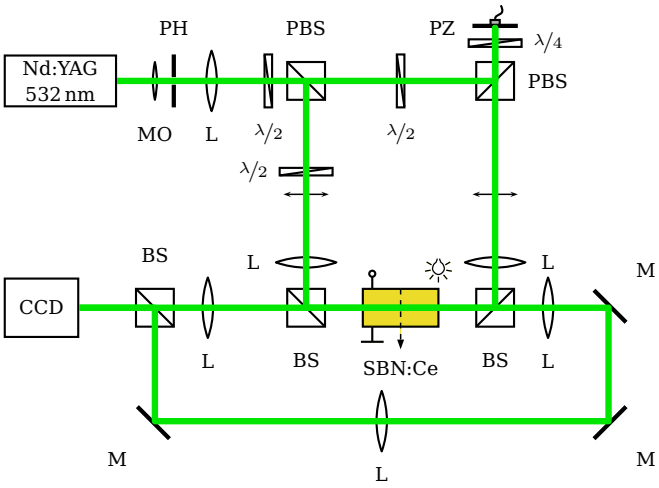

```
\begin{pspicture}(1,1.2)(6.2,5)
\pnode(1,4){SigIn}\pnode(3,4){BS}\pnode(3,5){LO}
\pnode(5,4){Det1}\pnode(3,2){Det2}
\begin{optexp}
  \optplane[compname=LO, angle=90](LO)
  \beamsplitter[compname=BS](SigIn)(BS)(Det2)
  \lens[abspos=0.5, compname=L2, n=2.1](Det2)(BS){$L_2$}
  \lens[abspos=0.5, compname=L1, n=2.1](Det1)(BS){$L_1$}
  \psset[optexp]{extnodealign=rel, extnode=r}
  \optdetector[compname=Det1](BS)(Det1){PD1}
  \optdetector[compname=Det2](BS)(Det2){PD2}
  \addtopsstyle{Beam}{beamwidth=0.2, fillstyle=solid,
                      fillcolor=green, opacity=0.2}
  \drawwidebeam(SigIn){BS}{L2}{Det2}
  \drawwidebeam[beaminsidefirst]{BS}{L1}{Det1}
  \newpsstyle{Beam}{linecolor=red, linestyle=dashed, linewidth=1.5\pslinewidth}
  \drawbeam{LO}{BS}{L1}{Det1}
  \drawbeam[beaminsidefirst]{BS}{L2}{Det2}
\end{optexp}
\cnodeput[framesep=5pt]([Xnodesep=0.5,
    offset=-0.5]\oenodeExt{Det1}|\oenodeExt{Det2}){M}{\rule{2.5mm}{1pt}}
\psset{arrows=<-, arrowscale=1.5, arrowinset=0}
\nccurve[angleA=90]{M}{\oenodeExt{Det1}}
\nccurve[angleA=180, angleB=-90]{M}{\oenodeExt{Det2}}
\end{pspicture}
```

Ex. 10.11



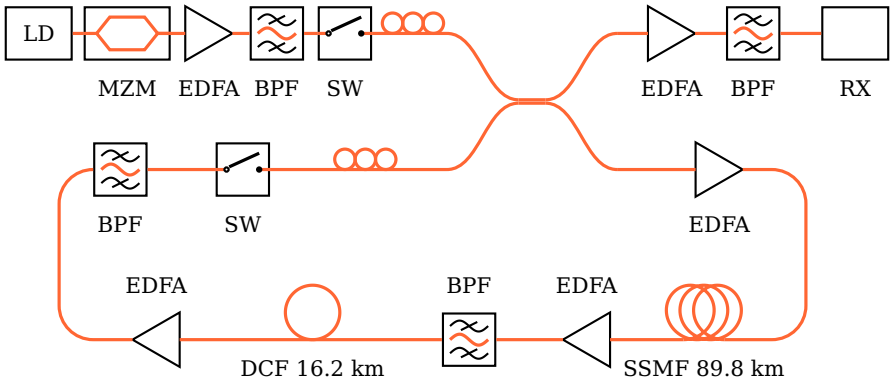

```
\begin{pspicture}(-0.2,0)(9,5)
\psset[optexp]{lens=1.2 0 1.2, n=1.72}
\addtopsstyle{Beam}{fillstyle=solid, fillcolor=green, opacity=0.3}
\pnode(2.4,1){BS1}\pnode([offset=3]BS1){M1}
\pnode([Xnodesep=5.5]M1){PP}\pnode(PP|BS1){BS2}
\begin{optexp}
  \optbox[label=0, position=start, optboxwidth=1.6]([Xnodesep=-1]BS1)(BS1){Nd:YAG}
  \beamsplitter[compname=BS](BS2)(BS1)(M1){BS}
  \optdipole[position=0.2, compname=LCLV, optdipolesize=0.28 1](BS1)(BS2){%
    \psframe[fillstyle=solid,fillcolor=black,dimen=outer](-0.14,-0.5)(-0.02,0.5)
    \psframe[fillstyle=solid,fillcolor=gray!50,dimen=outer](-0.02,-0.5)(0.14,0.5)
  }{LCLV}
  \optretplate(BS1)(M1){P}
  \mirror(BS1)(M1)(PP){M}
  \lens[position=0.2](M1)(PP){L}
  \pinhole(M1)(PP)
  \lens[position=0.2](PP)(M1){L}
  \pentaprism(M1)(PP)(BS2){PP}
  \beamsplitter(PP)(BS2)(BS1){BS}
  \doveprism[compname=Dove, position=0.27, n=2.3](BS2)(BS1){Dove}
  \lens[n=2.5](BS2)(BS1){L}
  \drawwidebeam[beamwidth=0.3]{1-3}
  \drawwidebeam[loadbeampoints]{3}{2}{4-}{3}
\end{optexp}
\end{pspicture}
```

Ex. 10.12



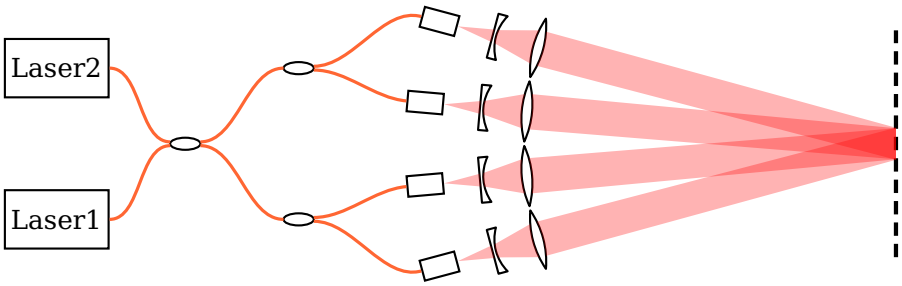

```
\begin{pspicture}(0,-0.2)(8.6,5.6)
\pnode(1.5,5){Laser}\pnode(4,5){PBS}\pnode(6.5,5){PBS2}\pnode(6.5,5.7){piezo}
\pnode(4,2){BSFwd}\pnode(6.5,2){BSBwd}\pnode(2,2){BS4f}\pnode(2,0.5){M4f3}
\pnode(8,2){M4f1}\pnode(8,0.5){M4f2}\pnode(1,2){CCD}
\psset{mirrorwidth=0.6, plateheight=0.7, outerheight=0.7, labeloffset=0.7,
    labelstyle=\scriptsize, lens=1.2 1.2 0.8, bssize=0.5}
\optbox[position=start, optboxsize=1.5 0.7, innerlabel]%
  (Laser)(PBS){\parbox{1.5cm}{\centering Nd:YAG\\ 532\,nm}}
\lens[lensheight=0.5, position=0.2](Laser)(PBS){MO}
\pinhole[position=0.3,labelangle=180](Laser)(PBS){PH}
\lens[position=0.5](Laser)(PBS){L}
\optretplate[position=0.8](Laser)(PBS){$\nicefrac{\lambda}{2}$}
\beamsplitter(Laser)(PBS)(BSFwd){PBS}
\optretplate[position=0.4](PBS)(BSFwd){$\nicefrac{\lambda}{2}$}
\polarization(PBS)(BSFwd)\polarization(PBS2)(BSBwd)
\lens[position=0.8](PBS)(BSFwd){L}
\optretplate(PBS)(PBS2){$\nicefrac{\lambda}{2}$}
\beamsplitter(PBS)(PBS2)(piezo){PBS}
\optretplate[abspos=0.5](PBS2)(piezo){$\nicefrac{\lambda}{4}$}
\mirror[mirrortype=piezo,labelangle=90](PBS2)(piezo)(PBS2){PZ}
\lens[position=0.8,labelangle=180](PBS2)(BSBwd){L}
\crystal[crystalsize=1 0.5, voltage, lamp, fillstyle=solid,
    fillcolor=yellow!90!black, labeloffset=0.8](BSFwd)(BSBwd){SBN:Ce}
\beamsplitter(PBS)(BSFwd)(BSBwd){BS}
\beamsplitter[labelangle=-90](PBS2)(BSBwd)(BSFwd){BS}
\mirror(BSBwd)(M4f1)(M4f2){M}\mirror(M4f1)(M4f2)(M4f3){M}
\lens[labelangle=180](M4f2)(M4f3){L}\mirror(M4f2)(M4f3)(BS4f){M}
\beamsplitter(M4f3)(BS4f)(CCD){BS}
\optbox[position=end, label=0, optboxwidth=1](BS4f)(CCD){CCD}
\lens[abspos=0.7](BS4f)(BSFwd){L}\lens[abspos=0.7](BSBwd)(M4f1){L}
\addtopsstyle{Beam}{linewidth=2\pslinewidth}
\drawbeam{1-6}{11-14}{13}{12}{15}{18}{16}{17}{25}{23}{24}
\drawbeam[beaminsidefirst]{6}{7}{10}{17}{16}{18}{26}{19-24}
\end{pspicture}
```

Ex. 10.13



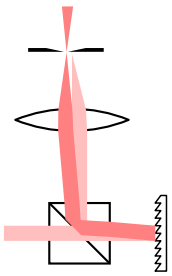

```
\begin{pspicture}(13,5.5)
\psset{usefiberstyle, optboxwidth=1, labelstyle=\footnotesize, fiber=none}
\pnode(1,5){LD}\pnode([Xnodesep=5.5]LD){CPLin1}
\pnode([offset=-2]CPLin1){CPLin2}\pnode([Xnodesep=2.5]CPLin1){CPLout1}
\pnode([Xnodesep=2.5]CPLin2){CPLout2}\pnode([Xnodesep=3]CPLout1){RX}
\optbox[position=start, label=0](LD)(CPLin1){LD}
\optmzm[abspos=0.8](LD)(CPLin1){MZM}
\optamp[abspos=2](LD)(CPLin1){EDFA}
\optfilter[abspos=3](LD)(CPLin1){BPF}
\optswitch[abspos=4](LD)(CPLin1){SW}
\polcontrol[abspos=5, fiber=out](LD)(CPLin1)
\drawfiber{1}{2}{3}{4}{5}{6}
\optcoupler[couplertype=none, fiber](CPLin1)(CPLin2)(CPLout1)(CPLout2)
\optamp[abspos=0.8, fiber=in](CPLout1)(RX){EDFA}
\optfilter[abspos=2](CPLout1)(RX){BPF}
\optbox[position=end](CPLout1)(RX){RX}
\drawfiber{8}{9}{10}
\pnode([Xnodesep=-0.5]LD|CPLin2){TL}\pnode(RX|TL){TR}
\pnode([offset=-2.5]TR){BR}\pnode(TL|BR){BL}
\optamp[fiber=i](CPLout2)(TR){EDFA}
\optfiber[label=0.3 . t, position=0.85](BL)(BR){SSMF 89.8~km}
\drawfiber[fiberstyle=angle, arm=1.2, linearc=0.5, startnode=N]{11}{12}
\optamp[position=0.3](BR)(BL){EDFA}
\optfilter[position=0.55, labelangle=180](BL)(BR){BPF}
\optfiber[fiberloops=1, label=0.3 . t, position=0.35](BL)(BR){DCF 16.2~km}
\optamp[position=0.85](BR)(BL){EDFA}
\optfilter[position=0.2](TL)(CPLin2){BPF}
\optswitch(TL)(CPLin2){SW}
\polcontrol[position=0.8, fiber=out](TL)(CPLin2)
\drawfiber{12}{13}{14}{15}{16}\drawfiber{17}{18}{19}
\drawfiber[fiberstyle=angle, arm=0.5, linearc=0.5, stopnode=1]{16}{17}
\end{pspicture}
```

Ex. 10.14



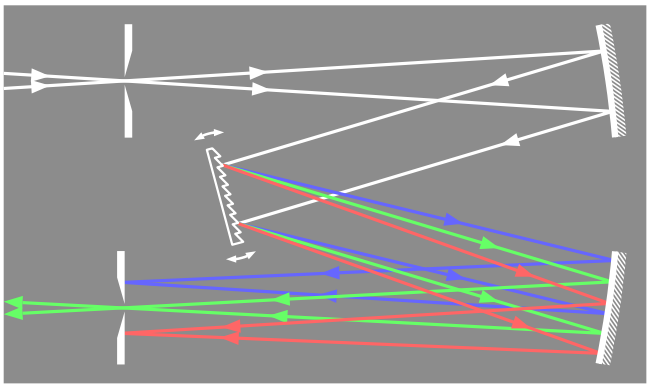

```
\begin{pspicture}(12, 4)
  \psset{labeloffset=0.3, fiber=none}
  \pnode(1.6,2){In}\pnode(12, 2){Ref}
  \optplate[linestyle=dashed, plateheight=3, position=1, compname=RefPlane](In)(Ref)
  \addtopsstyle{Beam}{linestyle=none, beamdiv=20, beaminside=false, ↩
      fillstyle=solid, fillcolor=red, opacity=0.3}
  \multido{\i=1+1, \ii=165+10}{4}{%
    \rput(Ref){\pnode(6;\ii){A\i}}
    \optbox[optboxsize=0.5 0.3, position=end, compname=Box\i, extnode=l](Ref)(A\i)
    \lens[lens=0 -0.6 0.6, abspos=0.5, compname=L\i](A\i)(Ref)
    \lens[lens=1.2 1.2 0.8, abspos=1.1, n=1.65, compname=LC\i](A\i)(Ref)
    \drawwidebeam{Box\i}{L\i}{LC\i}{RefPlane}
  }%
  \optcoupler[abspos=1](In)(In)(Ref)(Ref)
  \wdmsplitter[abspos=2.5, compshift=1](In)(Ref)(Ref)
  \wdmsplitter[abspos=2.5, compshift=-1](In)(Ref)(Ref)
  \optbox[compshift=-1, position=start, label=0](In)(Ref){Laser1}
  \optbox[compshift=1, position=start, label=0](In)(Ref){Laser2}
  \drawfiber{17}{14}{15}{Box1}\drawfiber{15}{Box2}
  \drawfiber{18}{14}{16}{Box3}\drawfiber{16}{Box4}
\end{pspicture}
```

Ex. 10.15



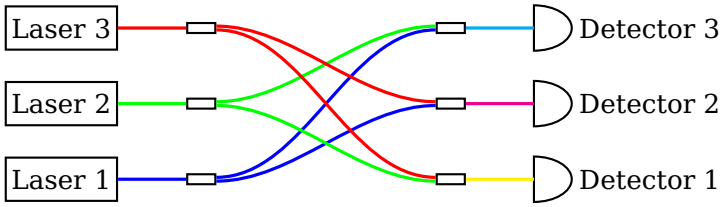
```
\begin{pspicture}(0.5,0)(2.2,4)
  \pnode(0,1){A}\pnode(1,1){BS}\pnode(2,1){G}\pnode(1,4){B}
  \beamsplitter(G)(BS)(B)
  \optgrating(BS)(G)(BS)
  \lens[lens=2 2 1.5, compshift=0.1, n=2.25](BS)(B)
  \pinhole[phwidth=0.05, innerheight=0.05, position=0.8, ↩
      compshift=0.18](BS)(B)
  \optplane[angle=90](B)
  \addtopsstyle{Beam}{linestyle=none, beamwidth=0.2, fillstyle=solid}
  \drawwidebeam[fillcolor=red!25!white](A){1-2}{1}{3-5}
  \drawwidebeam[fillcolor=red!50!white, beamangle=-5]{2}{1}{3-5}
\end{pspicture}
```

Ex. 10.16: Adapted from Wikipedia



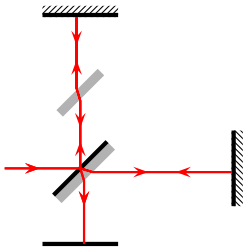
```
\begin{pspicture}(0,-1)(8.5,4)
\psframe[fillstyle=solid,fillcolor=gray!90,linestyle=none](0,-1)(8.5,4)
\addtopsstyle{OptComp}{linecolor=white}
\newpsstyle{Beam}{ArrowInside=->, linejoin=2, arrowscale=1.3, arrowinset=0}
\addtopsstyle{ExtendedMirror}{hatchcolor=white, hatchsep=0.5\pslinewidth}
\pnode(0,3){A}\pnode(8,3){B}\pnode(3,1.5){C}\pnode(8,0){D}\pnode(0,0){E}%
\psset{linewidth=1.5\pslinewidth, mirrorradius=13, mirrorwidth=1.5,
    gratingwidth=1.3, mirrortype=extended, phwidth=0.1, outerheight=1.5}%
\begin{optexp}
  \pinhole[position=0.2](A)(B)%
  \mirror(A)(B)(C)
  \optgrating[reverse, angle=15, variable](B)(C)(D)
  \mirror(C)(D)(E)
  \pinhole[position=0.8](D)(E)
  \drawwidebeam[ArrowInsidePos=0.3, beamwidth=0.2, linecolor=white,
      beamdiv=-7.15](A){1-3}
  \psset{loadbeampoints, savebeampoints=false}%
  \drawwidebeam[ArrowInsidePos=0.6, beamangle=3, linecolor=blue!60!white]{3-5}(E)
  \drawwidebeam[ArrowInsidePos=0.7, linecolor=green!60!white, arrows=->,
      ArrowInsideMinLength=2]{3-5}(E)
  \drawwidebeam[ArrowInsidePos=0.8, beamangle=-3, linecolor=red!60!white]{3-5}(E)
\end{optexp}
\end{pspicture}
```

Ex. 10.17



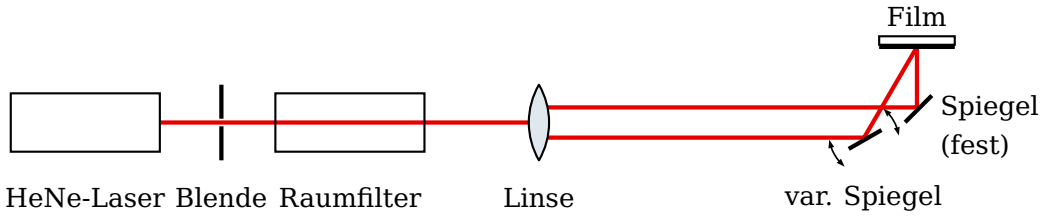

```
\begin{pspicture}(0,0.7)(9.4,3.3)
  \psset[optexp]{couplertype=rectangle, fiber=none, detsize=0.5 0.6}
  \multido{\i=1+1}{3}{%
    \optbox[optboxsize=1.5 0.6, innerlabel,position=start](1.5,\i)(7,\i){Laser \i}
    \wdmsplitter[position=0.2](1.5,\i)(7,\i)(7,\i)
    \wdmcoupler[position=0.8](1.5,\i)(1.5,\i)(7,\i)
    \optdetector[label=0.4 90 l](1.5,\i)(7,\i){Detector \i}
  }%
  \drawfiber[linecolor=blue, startnode=N]{1}{2}{7}
  \drawfiber[linecolor=blue]{2}{11}
  \drawfiber[linecolor=green, stopnode=1]{5}{6}{11}
  \drawfiber[linecolor=green, stopnode=2]{6}{3}
  \drawfiber[linecolor=red]{9}{10}{3}
  \drawfiber[linecolor=red, startnode=2]{10}{7}
  \drawfiber[linecolor=cyan]{11}{12}
  \drawfiber[linecolor=magenta]{7}{8}
  \drawfiber[linecolor=yellow]{3}{4}
\end{pspicture}
```

Ex. 10.18



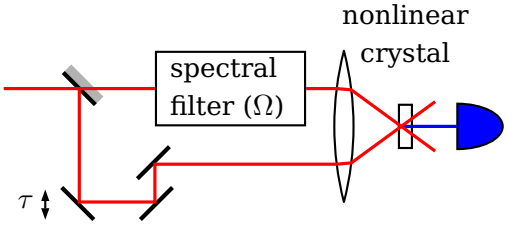
```
\begin{pspicture}(0,1)(3.2,4.2)
  \pnode(0,2){In}\pnode(1,2){G}\pnode(1,4){M1}
  \pnode(3,2){M2}\pnode(1,1){S}
  \mirror[mirrortype=semitrans](In)(G)(M1)
  \optbox[optboxwidth=0.15, angle=45,
      style=SemitransMirror](G)(M1)
  \mirror[mirrortype=extended](G)(M1)(G)
  \mirror[mirrortype=extended](G)(M2)(G)
  \optplate[position=end](G)(S)
  \newpsstyle{Beam}{linecolor=red, ArrowInside=->,
      ArrowInsidePos=0.4, arrowscale=1.3}
  \drawbeam(In){1}{4}{1}{5}
  \drawbeam(In){1-3}{2}{1}{5}
\end{pspicture}
```

Ex. 10.19: Adapted from semibyte.de



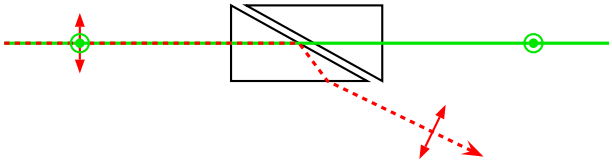

```
\begin{pspicture}(13.6,2.6)
  \psset[optexp]{labeloffset=1, mirrorwidth=0.5, optboxwidth=2}
  \pnode(2,1){Laser}\pnode(12,2){Film}\pnode(Film|Laser){Ref}
  \begin{optexp}
    \optbox[position=start](Laser)(Ref){HeNe-Laser}
    \pinhole[abspos=0.8](Laser)(Ref){Blende}
    \optbox[abspos=2.5](Laser)(Ref){Raumfilter}
    \lens[fillstyle=solid, fillcolor={[rgb]{0.87,0.91,0.93}}](Laser)(Ref){Linse}
    \mirror[label=1 30]([offset=0.2]Laser)([offset=0.2]Ref)(Film){%
      \begin{tabular}{l}Spiegel\\(fest)\end{tabular}}
    \opttripole[label=0.4](Ref)(Film)(Ref){%
      \psframe(0.5,1\pslinewidth)(-0.5,5\pslinewidth)
      \psline[linewidth=2\pslinewidth](0.5,0)(-0.5,0)
    }{Film}
    \newpsstyle{Beam}{linewidth=2\pslinewidth, linecolor=red!90!black,
       beamalign=absolute}
    \drawbeam{1-4}
    \drawbeam[beampos=0.2]{4-6}
    \mirror[variable, label=0.8 -30]([offset=-0.2]Laser)([offset=-0.2,
       Xnodesep=-0.7]Ref)(Film){var. Spiegel}
    \drawbeam[beampos=-0.2]{4}{7}{6}
  \end{optexp}
\end{pspicture}
```

Ex. 10.20



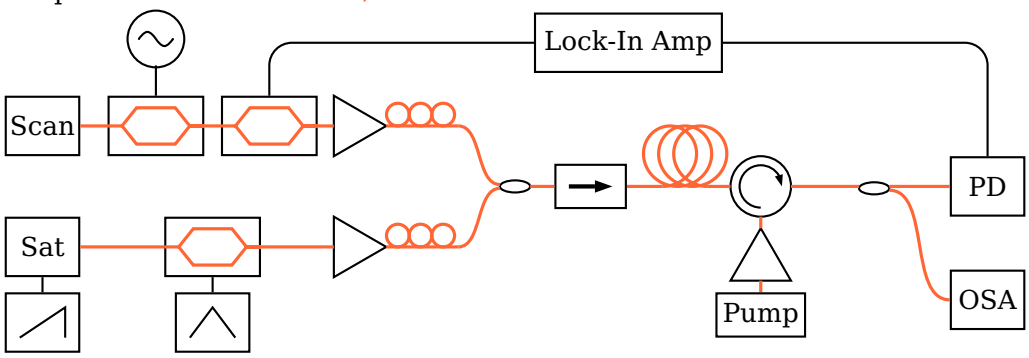

```
\begin{pspicture}(0,0.2)(6.6,3.1)
  \psset{mirrorwidth=0.6, useNA=false, beaminsidefirst}
  \pnode(0,2){In}\pnode(1,2){BS}\pnode(1,0.5){M1}\pnode(2,1){M2}\pnode(6,1.5){Det}
  \mirror[mirrortype=semitrans, n=1](In)(BS)(M1)
  \mirror[label=0.6 180 . absolute, compname=M](BS)(M1)(M2|M1)
  \mirror(M1)(M2|M1)(M2)
  \mirror(M2|M1)(M2)([Xnodesep=1]M2)
  \optbox[optboxsize=2 1, abspos=2, label=0,
      allowbeaminside=false](BS)([offset=0.5]Det){%
    \parbox{1.6\psxunit}{spectral\\filter ($\Omega$)}}
  \lens[lens=4 4 2, n=3.3]([Xnodesep=-3]Det)(Det)
  \optbox[optboxsize=0.2 0.6, n=1, label=-1.2](\oenodeOut{6})(Det){%
    \begin{tabular}{@{}c@{}}nonlinear\\ crystal\end{tabular}}
  \optplate[linestyle=none](\oenodeOut{7})(Det)
  \optdetector[detsize=0.6 0.6, fillstyle=solid,fillcolor=blue](\oenodeOut{6})(Det)
  \newpsstyle{Beam}{linewidth=1.5\pslinewidth}
  \drawbeam[linecolor=blue]{7-9}
  \addtopsstyle{Beam}{linecolor=red}
  \drawbeam(In){1-4}{6-8}
  \drawbeam{1}{5-8}
  \rput[r](\oenodeLabel{M}){$\tau$
    \psline[arrows=<->, arrowinset=0](0.15,0.25)(0.15,-0.15)}
\end{pspicture}
```

Ex. 10.21



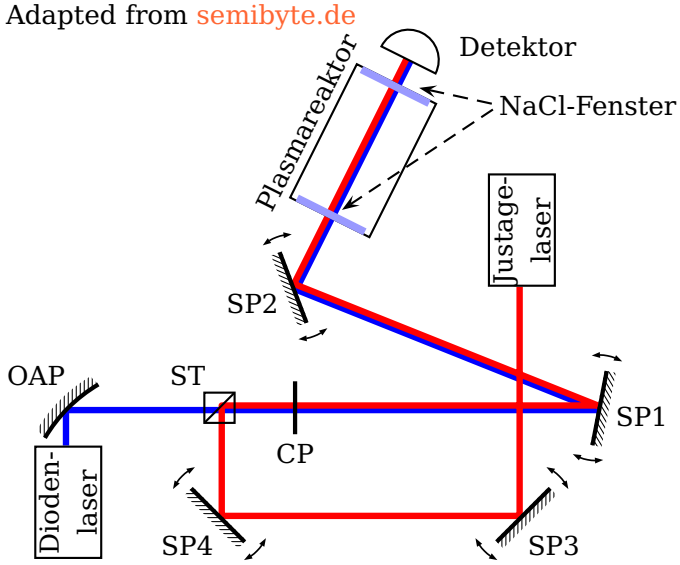
```
\begin{pspicture}(-1,-0.5)(8,1.5)
  \newpsobject{perppol}{polarization}{poltype=perp, linecolor=green!90!black,
      polsize=1}
  \newpsobject{parallelpol}{polarization}{poltype=parallel, linecolor=red,
      polsize=0.8}
  \newpsstyle{Polarization}{arrowinset=0, arrowscale=1.2}
  \pnode(0,1){A}\pnode(8,1){B}\pnode(4,-0.5){C}
  \glanthompson[glanthompsongap=0.2, glanthompsonsize=2 1](A)(B)
  \optplane(B)\optplane[angle=90](C)
  \drawbeam(A){1-2}
  \drawbeam[linestyle=dashed, dash=2pt 2pt, linecolor=red, arrowscale=1.5,
      arrows=->](A){1}{3}
  \parallelpol[abspos=1](A)(B)
  \perppol[abspos=1](A)(B)
  \nodexn{(\oenodeBeam{})-1.5(!\oeBeamVec{})}{D}
  \perppol[abspos=7](A)(B)
  \parallelpol(D)(\oenodeBeam{})
\end{pspicture}
```

Ex. 10.22: Adapted from DOI:10.1364/OL.37.000930



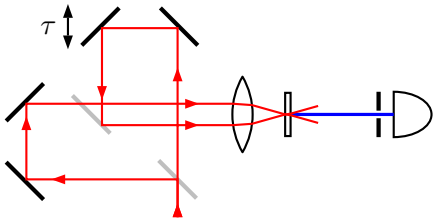

```
\begin{pspicture}(0,0.2)(13.5,4.7)
\psset[optexp]{optboxsize=1 0.8, usefiberstyle}
\pnode(1,3.2){Scan}\pnode([offset=-1.6]Scan){Sat}
\pnode([Xnodesep=5]Scan){ScanCpl}\pnode(ScanCpl|Sat){SatCpl}
\pnode([Xnodesep=1.5,offset=-0.8]ScanCpl){Cpl}\pnode([Xnodesep=5]Cpl){PD}
\pnode([Xnodesep=2.5, offset=-1.4]Cpl){Pump}\psset[optexp]{fiber=none}
\optbox[position=start, label=0](Scan)(ScanCpl){Scan}
\optmzm[abspos=1, compname=M1, extnode=t](Scan)(ScanCpl)
\optmzm[abspos=2.5, compname=M2, extnode=t](Scan)(ScanCpl)
\optamp[abspos=3.7](Scan)(ScanCpl)
\polcontrol[abspos=4.5, fiber=o](Scan)(ScanCpl)
\drawfiber{1}{2}{3}{4}{5}
\optbox[position=start, compname=S, extnode=b, label=0](Sat)(SatCpl){Sat}
\optmzm[abspos=1.75, compname=M3, extnode=b](Sat)(SatCpl)
\optamp[abspos=3.7](Sat)(SatCpl)
\polcontrol[abspos=4.5, fiber=o](Sat)(SatCpl)
\drawfiber{6}{7}{8}{9}
\wdmcoupler[fiber=i](ScanCpl)(SatCpl)(Cpl)
\optisolator[position=0.1](Cpl)(Pump|Cpl)
\optfiber[position=0.6](Cpl)(Pump|Cpl)
\addtopsstyle{OptCircArrow}{arrows=<-}
\optcirculator[optcircangle=0 -90](Cpl)(PD)(Pump)
\wdmsplitter[position=0.8, coupleralign=t](Cpl)(PD)
\optbox[position=end, label=0, compname=Pd, extnode=t](Cpl)(PD){PD}
\optbox[position=end, label=0, compshift=-1.5](Cpl)(PD){OSA}
\optamp[fiber](Pump)(\oenodeIfc{2}{13})
\optbox[position=start, optboxsize=0.6 1.2, label=0 . . ↩
    absolute](Pump)(Pump|Cpl){Pump}
\drawfiber{10}{11}{12}{13}{14}{15}\drawfiber{14}{16}
% electrical stuff
\newpsstyle{Fiber}{}\psset[optexp]{label=0}
\optbox[compshift=0.7, optboxsize=2.5 ↩
    0.8](\oenodeExt{M2})(\oenodeExt{Pd}|\oenodeExt{M2}){Lock-In Amp}
\ncangle[angleA=90, angleB=180, linearc=0.3]{\oenodeExt{M2}}{\oenodeIn{}}
\ncangle[angleB=90, linearc=0.3]{\oenodeOut{}}{\oenodeExt{Pd}}
\psset[optexp]{fiber=all, optboxsize=0.8 1, position=start}
\optbox(\oenodeExt{S}|Pump)(\oenodeExt{S}){\psline(-0.3,-0.2)(0.3,0.2)(0.3,-0.2)}
\optbox(\oenodeExt{M3}|Pump)(\oenodeExt{M3}){\psline(-0.3,-0.2)(0,0.2)(0.3,-0.2)}
\oscillator[output=none, unit=0.75, ↩
    linewidth=0.666\pslinewidth]([offset=1]\oenodeExt{M1}){}{}
\psline(\oenodeExt{M1})([offset=0.4]\oenodeExt{M1})
\end{pspicture}
```

Ex. 10.23: Adapted from semibyte.de



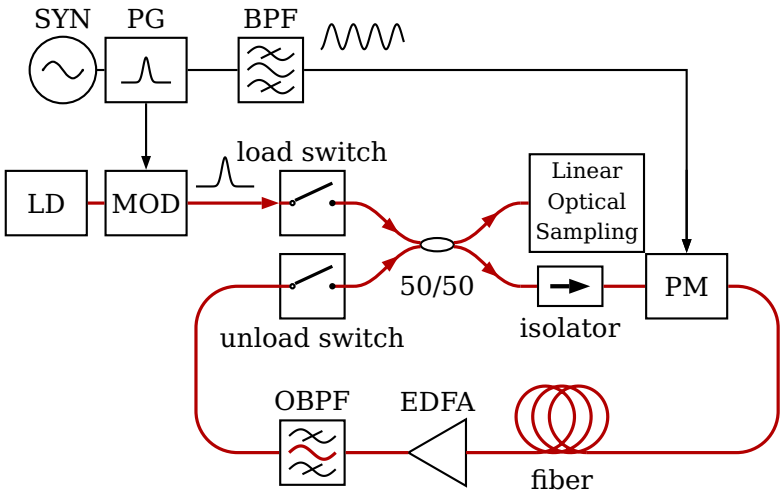

```
\begin{pspicture}(0.2,0.5)(9,7.4)
  \pnode(1,2){DL}\pnode([offset=0.5]DL){OAP}\pnode([Xnodesep=2]OAP){ST}
  \pnode([offset=-1.4]ST){SP4}\pnode([Xnodesep=4]SP4){SP3}
  \pnode([offset=3]SP3){JL}\pnode([Xnodesep=1]SP3|ST){SP1}
  \pnode([Xnodesep=1]ST|JL){SP2}\pnode([Xnodesep=1.5,offset=3]SP2){Det}
  \psset[optexp]{optboxsize=1.5 0.9, labeloffset=0.6}
  \begin{optexp}
    \optbox[position=start, innerlabel](DL)(OAP){\parbox{1.5\psunit}{\centering
       Dioden-\\laser}}
    \psset[optexp]{mirrortype=extended}
    \mirror[mirrorradius=2](DL)(OAP)(ST){OAP}
    \psset[optexp]{variable}
    \beamsplitter[bssize=0.4, labelangle=-45](SP4)(ST)(SP1){ST}
    \optplate[abspos=1, plateheight=0.7](ST)(SP1){CP}
    \mirror(ST)(SP1)(SP2){SP1}\mirror(SP1)(SP2)(Det){SP2}
    \optdipole[label=-0.7 . b relative, optdipolesize=2, position=0.6](SP2)(Det){%
      \psframe[dimen=outer](-1,-0.6)(1,0.6)
      \psset{fillstyle=solid, fillcolor=blue!40, linestyle=none}
      \psframe(-1,-0.5)(-0.9,0.5)\psframe(0.9,-0.5)(1,0.5)}{Plasmareaktor}
    \optdetector[label=. 30 l](SP2)(Det){Detektor}
    \optbox[position=end, innerlabel](SP3)(JL){\parbox{1.5\psunit}{\centering
       Justage-\\laser}}
    \mirror(SP4)(SP3)(JL){SP3}\mirror(ST)(SP4)(SP3){SP4}
    \psset{linewidth=2\pslinewidth}
    \drawbeam[linecolor=blue, beampos=0.03]{1-2}
    \drawbeam[beamalign=abs, loadbeampoints, linecolor=blue]{2-8}
    \drawbeam[linecolor=red, beampos=-0.03]{9-11}{3-8}
  \end{optexp}
  \pnode([Xnodesep=1.2,offset=-0.6]\oenodeIn{8}){NaCl}
  \psset{linestyle=dashed, arrows=->, arrowscale=1.5}
  \ncline[nodesepA=0.1, nodesepB=0.2]{NaCl}{\oenodeIn{7}}
  \ncline[nodesepA=0.1, nodesepB=0.3]{NaCl}{\oenodeOut{7}}
  \rput[l](NaCl){NaCl-Fenster}
\end{pspicture}
```

Ex. 10.24: Adapted from DOI:10.1364/AOP.1.000308, Fig. 8

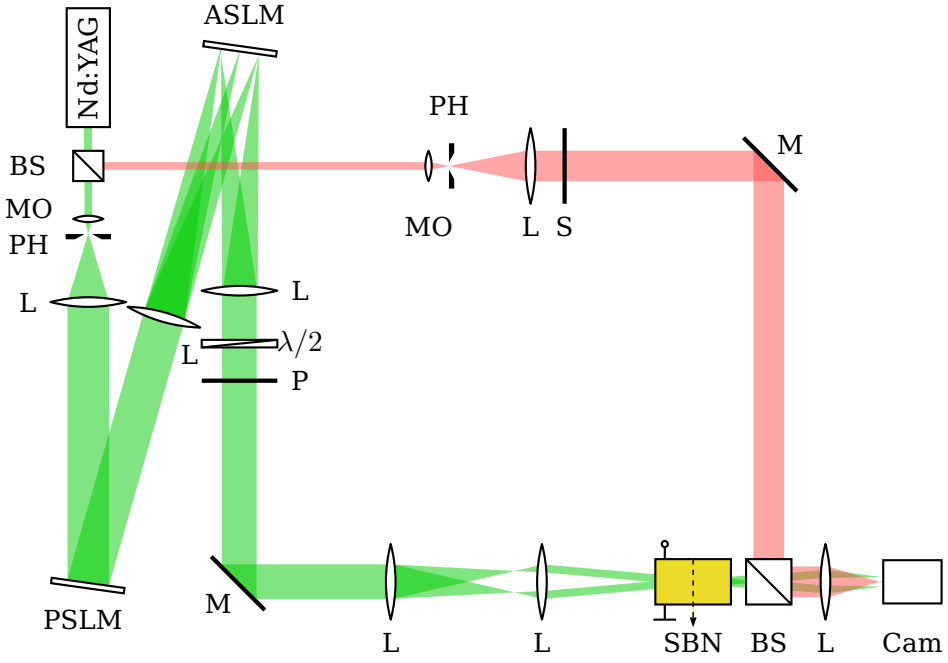



```
\begin{pspicture}(5.7,2.8)
\psset{mirrorwidth=0.7, bsstyle=plate, bssize=0.5, arrowinset=0, arrowscale=1.2}
\newpsobject{splitterplate}{beamsplitter}{linewidth=2\pslinewidth,
    linecolor=gray!50}
\pnode(2.3,0){In}\pnode([offset=0.5]In){STM1}
\pnode([Xnodesep=-2]STM1){M1}\pnode([offset=1]M1){M2}
\pnode([offset=2]STM1){M3}\pnode([Xnodesep=-1]M3){M4}
\pnode(M4|M2){STM2}
\splitterplate(In)(STM1)(M1)
\mirror(STM1)(M1)(M2)
\mirror(M1)(M2)(M3|M2)
\mirror(STM1)(M3)(M4)
\mirror[label=0.6 180 . absolute](M3)(M4)(STM2)
\rput[r](\oenodeLabel{}){$\tau$
  \psline[arrows=<->](0.15, 0.4)(0.15, -0.2)}
\splitterplate[compoffset=0.2, compname=STM2](M4)(STM2)([Xnodesep=1]STM2)
\pnode(\oenodeCenter{STM2}){STM2C}
\pnode([Xnodesep=4]STM2C){Out}
\lens[n=2](STM2C)(Out)
\optbox[position=0.65, optboxsize=0.1 0.6](STM2C)(Out)
\optplate[linestyle=none, position=0.75](STM2C)(Out)
\pinhole[position=0.95, outerheight=0.6](STM2C)(Out)
\optdetector[detsize=0.5 0.6](STM2C)(Out)
\drawbeam[linecolor=blue]{8-}
\newpsstyle{Beam}{ArrowInside=->, linecolor=red, ArrowInsideMinLength=1, arrows=>-}
\drawbeam[ArrowInsidePos=0.835](In){1-3}{7-9}
\drawbeam[ArrowInsidePos=0.74](In){1}{4-9}
\end{pspicture}
```

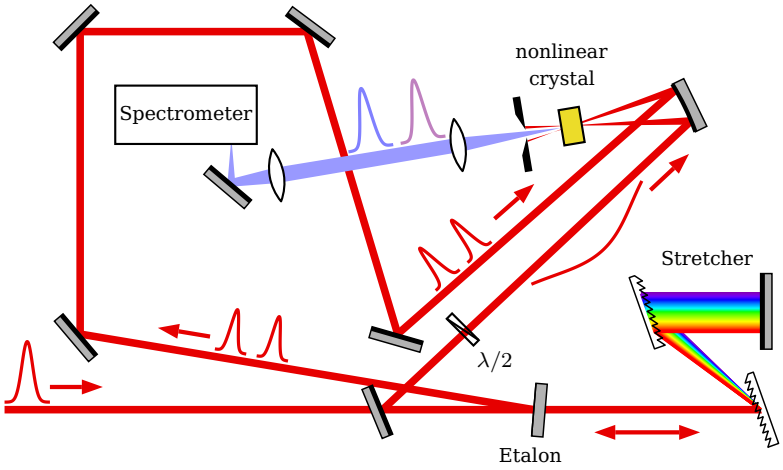
Ex. 10.25



Ex. 10.26: Adapted from DOI:10.1364/OL.37.000797



Ex. 10.27: Adapted from DOI:10.1364/AOP.1.000308, Fig. 37



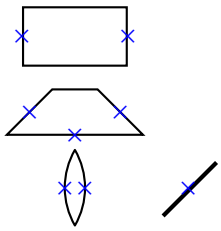
11. Additional information

11.1. Internal component structure

showifcnodes=true, false default: false

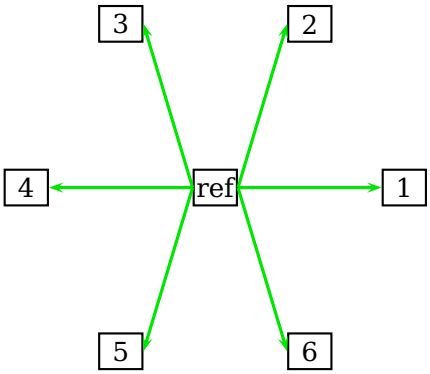
IfcNodeStyle⟨*psstyle*⟩ default: dotstyle=x, dotscale=1.5, linecolor=blue

Every component consists internally of interfaces which are defined by a node on the optical axis, a plane vector or a curvature radius, and by its optical characteristics (reflective or transmittive, or both e.g. for \beamsplitter). The interface nodes can be visualized for each component with the showifcnodes parameter.



```
\begin{pspicture}(4,3)
  \psset{showifcnodes}
  \optbox(0,2.5)(4,2.5)
  \doveprism(0,1.5)(4,1.5)
  \lens(0,0.5)(4,0.5)
  \mirror(0,0.5)(3.5,0.5)(3.5,3)
\end{pspicture}
```

\draw*beam calculates the distance between the interface nodes of two components and connects the two nearest to each other.




```
\begin{pspicture}(-3,-3)(3,3)
  \psset{optboxsize=0.6 0.5, labeloffset=0}
  \multido{\i=0+60,\ii=1+1}{6}{%
    \pnode(2.5;\i){A}
    \optbox([Xnodesep=-1]A)([Xnodesep=1]A){\ii}
  }
  \optbox[compname=ref](-1,0)(1,0){ref}
  \addtopsstyle{Beam}{arrows=->}
  \multido{\i=1+1}{6}{\drawbeam{ref}{\i}}
\end{pspicture}
```

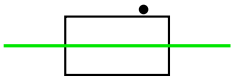
11.2. Overview of special nodes

In this section you find overviews of all special nodes which each component provides. The possible positions for external and rotation reference nodes, and the interface nodes.

11.2.1. External and rotation reference nodes

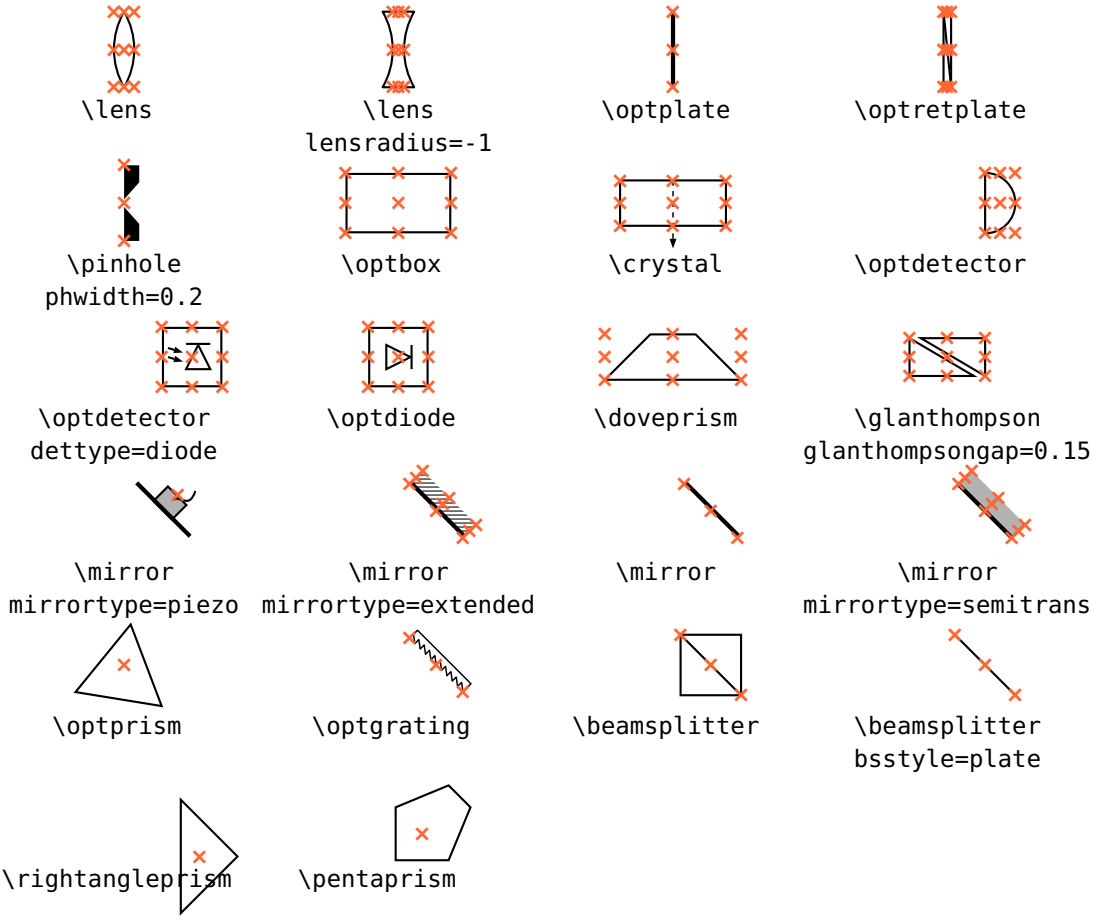
Here, you find all possible external (Sec. 7.5) and rotation reference nodes (Sec. 7.7) for all components. Displayed are all possible combinations of `t`, `c`, or `b` with `l`, `c`, or `r`. Not all components support all combinations, e.g. a plain `\mirror` ignores `t` and `b` and an `\optplate` ignores `l` and `r`.

Instead of these values also fractional parts can be used: `t` corresponds to $+1 \cdot \Delta y$, `b` corresponds to $-1 \cdot \Delta y$, `l` corresponds to $-1 \cdot \Delta x$, and `r` corresponds to $+1 \cdot \Delta x$. In the following example the point $(0.5\Delta x, 1.2\Delta y)$ is provided as external node.

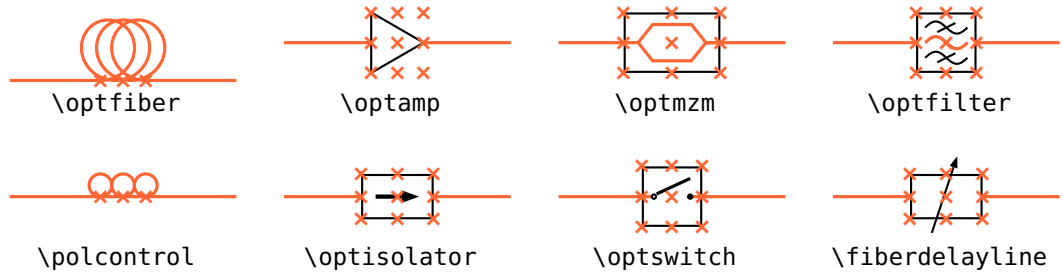


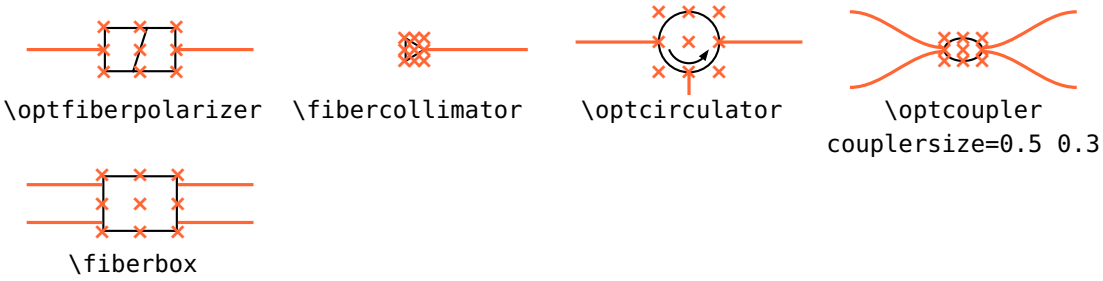
```
\begin{pspicture}(3,1)
  \optbox[extnode={0.5,1.2}, beam](0,0.5)(3,0.5)
  \psdot(\oenodeExt{})
\end{pspicture}
```

Free-ray components



Fiber components

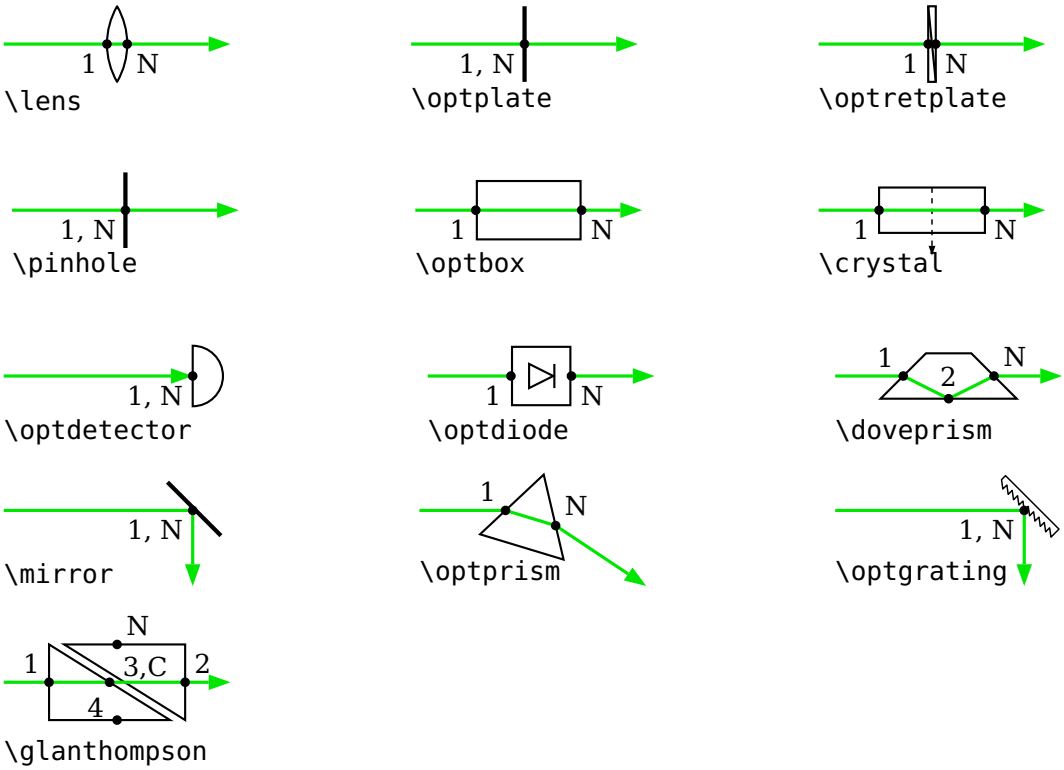


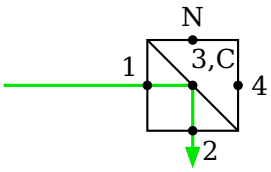


11.2.2. Interface nodes

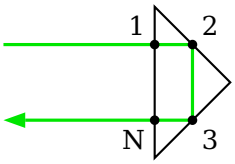
Here, all available components and their interface nodes are listed. If a node is labeled as “1, N”, it means that both nodes are equal.

Free-ray components

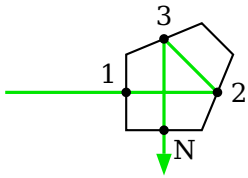




\beamsplitter

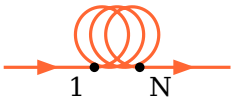


\rightangleprism

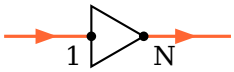


\pentaprism

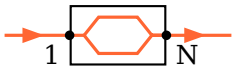
Fiber components



\optfiber



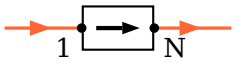
\optamp



\optmzm

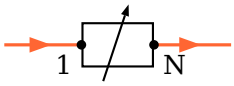
\optfilter

\polcontrol

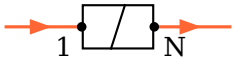


\optisolator

\optswitch

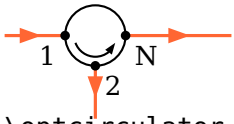


\fiberdelayline

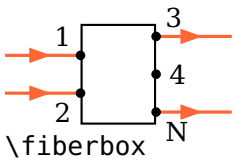


\optfiberpolarizer

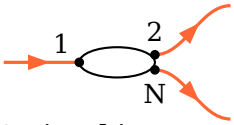
\fibercollimator



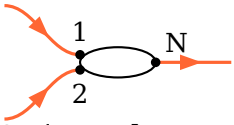
\optcirculator



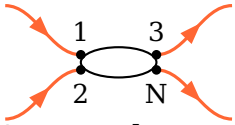
\fiberbox



\wdmsplitter



\wdmcoupler



\optcoupler

11.3. Backward compatibility

In this section you find a comprehensive listing of the behaviour which changed in a version compared to the previous one and which is not backward compatible. Here you find also information on what you must note when upgrading and how you must change old documents to work correctly with the newer version.

11.3.1. Version 3.3

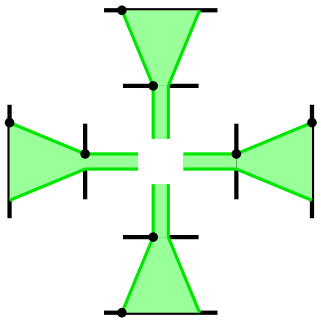
\opttripole Changed the reference angle for the label of the \opttripole to be consistent with the other tripoles.

Beam nodes

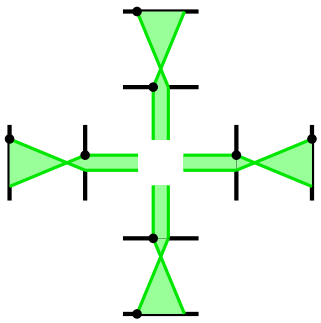
Before version 3.3 the beam nodes \oenodeBeamUp and \oenodeBeamLow were not swapped correctly and the orientation of “upper” and “lower” was only relative to the beam direction (equivalent to extnodealign=relative).

This could lead to problems when using loadbeampoints with beamdiv. In that case the sign of beamdiv was not unambiguously associated with contracting or expanding beam.

As of version 3.3 the beam nodes are arranged according to extnodealign=absolute and the association of negative beamdiv with a contracting and positive beamdiv with an expanding beam is always valid. In the following two examples all \oenodeBeamUp are marked.



```
\begin{pspicture}(-2,-2)(2,2)
  \addtopsstyle{Beam}{fillstyle=solid, fillcolor=green!40}
  \multido{\i=0+90,\ia=1+2,\ib=2+2}{4}{%
    \optplate[position=end](0.3;\i)(1;\i)
    \optplate[position=end, plateheight=1.5](1;\i)(2;\i)
    \drawwidebeam[beamwidth=0.2](0.3;\i){\ia}
    \pnode(\oenodeBeamUp{}){Up}
    \drawwidebeam[loadbeampoints, beamdiv=45]{\ia-\ib}
    \psdot(Up)\psdot(\oenodeBeamUp{})
  }
\end{pspicture}
```



```
\begin{pspicture}(-2,-2)(2,2)
  \addtopsstyle{Beam}{fillstyle=solid, fillcolor=green!40}
  \multido{\i=0+90,\ia=1+2,\ib=2+2}{4}{%
    \optplate[position=end](0.3;\i)(1;\i)
    \optplate[position=end](1;\i)(2;\i)
    \drawwidebeam[beamwidth=0.2](0.3;\i){\ia}
    \pnode(\oenodeBeamUp{}){Up}
    \drawwidebeam[loadbeampoints, beamdiv=-45]{\ia-\ib}
    \psdot(Up)\psdot(\oenodeBeamUp{})
  }
\end{pspicture}
```

11.3.2. Version 3.0

The changes in version 3.0 compared to version 2.1 which are not backward compatible.

General parameters

`namingscheme`=old, new default: new

In version 2.1 special component nodes had to be accessed by their explicit name. If you need the old naming scheme, because you accessed internal nodes directly, then you must set `namingscheme=old`. Since version 3.0 explicit macros are provided to access all special component nodes, so that the actual naming scheme does not matter, see Sec. 7.

`endbox` `angle` `rotateref` These options now affect all components.

`extnode` Some components now provide more possible positions for `extnode`. `\optdetector` had only one possible external node, which was accessible for any value of `extnode`, now this node can be accessed only with `extnode=r`. All other values give different node positions.

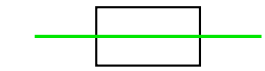
`\optbox` An `\optbox` does always have the input and output nodes at the respective interfaces, whereas in version 2.1 the input and output nodes where at the same position for `endbox=true`.

`\fibercollimator` The `\fibercollimator` has only a fiber connection by default, the beam must be drawn manually or with `beam` option.

Beam connections

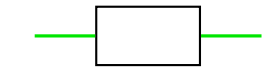
In version 2.1, beam connections could be internal, that is the connections were specified together with the compent (`beam` or `conn`), or external with an additional call of `\drawbeam`. The main reason for internal connections was the aesthetic aspect of drawing the beam behind the component, but also convenience played a role. Drawing the beam behind the component is possible with layering (Sec. 8.7) since 3.0.

Some of the variants of internal beam connections can be used also in 3.0:



conn=o-i, conn=i-o

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \optbox[beam](A)(B)
\end{pspicture}
```



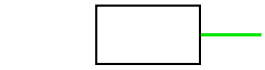
conn=o-o

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \optbox[beam, beaminside=false](A)(B)
\end{pspicture}
```

Other internal connections need an additional \drawbeam call, and if you want to move the beam behind the component, the code must be wrapped in a optexp environment. Except for this, external connection are equivalent:

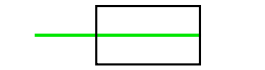
conn=o-, conn=a-

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \optbox(A)(B)
  \drawbeam(A){}
\end{pspicture}
```



conn=-o, conn=-b

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \optbox(A)(B)
  \drawbeam{}(B)
\end{pspicture}
```



conn=i-, A-

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \begin{optexp}
    \optbox(A)(B)
    \drawbeam[beaminsidelast](A){}
  \end{optexp}
\end{pspicture}
```

conn=-i, conn=-B

```
\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \begin{optexp}
    \optbox(A)(B)
    \drawbeam[beaminsidefirst]{}(B)
  \end{optexp}
\end{pspicture}
```

Documentation index

E

F

G

P

R

S

A. Revision history

This revision history is a list of changes relevant to users of this package. Changes of a more technical nature which do not affect the user interface or the behavior of the package are not included in the list. If an entry in the revision history states that a feature has been *improved* or *extended*, this indicates a modification which either does not affect the syntax and behavior of the package or is syntactically backwards compatible (such as the addition of an optional argument to an existing command). Entries stating that a feature has been *deprecated*, *modified*, *renamed*, or *removed* demand attention. They indicate a modification which may require changes to existing documents. The numbers on the right indicate the relevant section of this manual, more information regarding backward compatibility can be found in Sec. 11.3.

3.6 2013-03-20

Added option `compoffset` 3.3
Added option `beampathskip` 8.2.3
Added option `innercompalign` 3.3
Fixed some bugs in the raytracing and beam code.

3.5 2013-02-22

Added option `filterangle` 6.1
Fixed wrong output fiber style in `\optcoupler`
Fixed strange Postscript error which occured with some interpreters.

3.4 2013-02-03

Fixed bugs when using `fillstyle` for some components
Extended option `switchsize` 5.6
Modified coupler center node to be on the base line 7.3
Extended error checking for `\drawbeam` and `\drawfiber` 8.1
Fixed some bugs in the connection code

3.3a 2012-09-18

Fixed bug which was exposed by an update of pst-node.
Fixed trailing spaces.

3.3 2012-08-17

Extended `\opttripole` and `\optdipole` 9
Modified reference angle for label of `\opttripole` 11.3.1
Added option `optdipolesize` to support two interfaces for `\optdipole` . 9
Added option `gratingalign` 4.14
Added option `forcebeaminside` 8.2.4
Extended option `mirrorradius` 4.12
Improved `\drawbeam` and `\drawwidebeam` to be able to use `ArrowInside` without linestyle 8.2.7
Fixed orientation of `\oenodeBeamUp` and `\oenodeBeamLow` 7.8
Fixed a bug related to `beaminside` and ambiguous components.
Fixed some trailing spaces
Added more examples 10

3.2 2012-07-26

Added component `\glanthompson` 4.10
Added access to beam vectors with `\oeBeamVec`, `\oeBeamVecUp` and `\oeBeamVecLow` 7.9
Fixed wrong computation of node distance in `\fiberbox`
Fixed bug in `\wdmsplitter` for `coupleralign=b` and `couplertype=none`
Fixed bug when filling components placed with `position=end/start`
Added value `absolute` to parameter `labelref` 3.1
Added examples 10

3.1 2012-07-17

Added component `\fiberbox` 5.11
Extended connection macros to not required curly braces around node parenthesis any more 7.1
Modified `\fibercollimator` to have `allowbeaminside=false` by default 6.2
Extended fiber couplers to allow using only two nodes 5.10
Fixed a bug concerning node expressions with `\drawfiber`

3.0 2012-07-09

2.1 2009-11-05

2.0 2008-07-27

1.2 2008-06-17

1.1 2007-09-06

1.0 2007-07-18

First CTAN version