PSTricks

pst-optic

Lenses and Mirrors; v.1.00

February 10, 2009

Documentation by
Herbert Voß

Package author(s):
Manuel Luque
Herbert Voß

Contents

`pst-optic` loads by default the following packages: `pstricks`, `pst-node`, `pst-plot`, `pst-3d`, `pst-grad`, `pst-math`, `multido`, and `pst-xke`. All should be already part of your local TEX installation. If not, or in case of having older versions, go to `http://www.CTAN.org/` and load the newest version.

Thanks to:
Jean-Côme Charpentier, Arnaud Schmittbuhl, and Timothy Van Zandt

1 General Options

All options are by default document wide valid but not supported by all macros. Table 1 shows the general ones. Others are shown in Table 2 and 4.

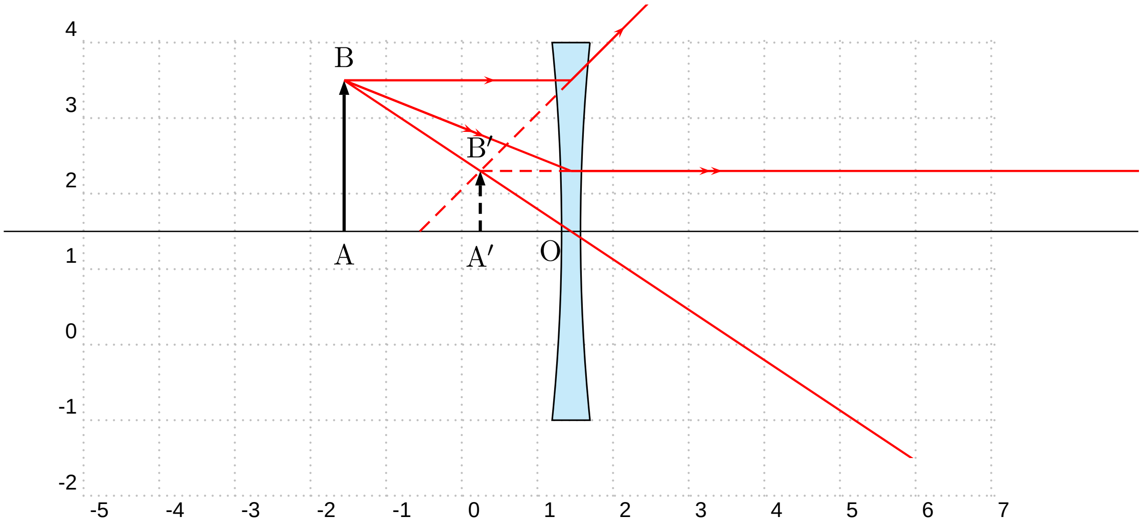
Table 1: General options and the defaults

Option	*Name*	*Default*
Left value of the picture in cm	`xLeft`	-7.5
Right value of the picture in cm	`xRight`	7.5
Lowest value of the picture in cm	`xBottom`	-3
Highest value of the picture in cm	`xTop`	3
x-Offset	`XO`	0
y-Offset	`YO`	0
Node A as string	`nameA`	A
Angle A in degrees	`spotA`	270
Node B as string	`nameB`	B
Angle B in degrees	`spotB`	270
Node F as string	`nameF`	F
Angle F in degrees	`spotF`	270
Node O as string	`nameO`	O
Angle O in degrees	`spotO`	225
Node A' as string	`nameAi`	A'
Angle A' in degrees	`spotAi`	90
Node B' as string	`nameBi`	B'
Angle B' in degrees	`spotBi`	270
Node F' as string	`nameFi`	B'
Angle F' in degrees	`spotFi`	270
Ray color	`rayColor`	black

`\pst-optic` puts the lens and mirror macros in an own `pspicture` environment. The star version enables the clipping option of `pstricks`:

```
\begin{pspicture}*(xLeft,yBottom)(xRight,yTop)
 \lens[focus=2,OA=-3,AB=1,XO=0,YO=0,xLeft=-7.5,xRight=7.5,yBottom=-3,yTop=3]
\end{pspicture}
```

If you need other values for the `pspicture` environment, then use the `\rput` command to place the macro at any position.




```
\begin{pspicture}[showgrid=true](-5,-2.2)(7,4)
\rput(1.5,1.5){%
 \lens[lensType=DVG,lensGlass=true,lensWidth=0.5,rayColor=red,
     focus=-2,AB=2,spotAi=270,spotBi=90]}
\end{pspicture}
```

1.1 \resetOpticOptions

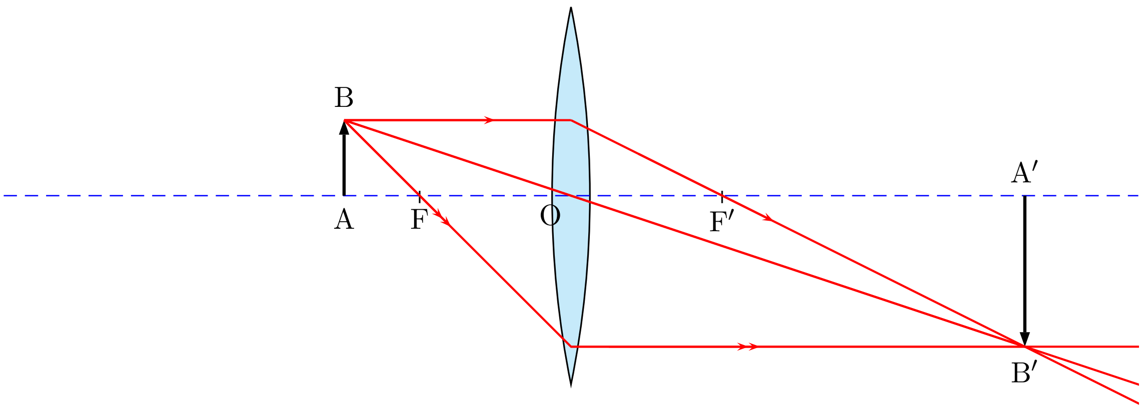
The Macro `\resetOpticOptions` resets all `pst-optic` options to the default value.

1.2 Optical axis line style

`pst-optic` defines a line style `opticalAxis` with the predefines values of:

```
\newpsstyle{opticalAxis}{linewidth=0.5pt,linecolor=black,linestyle=solid}
```

It can be overwritten in the same way with `\newpsstyle`.




```
\newpsstyle{opticalAxis}{linewidth=0.5pt,linecolor=blue,linestyle=dashed}
\lens
```

2 Lenses

There are macros for the convergent and divergent lens

```
\lens [Options]
\lensCVG [Options] % Convergent (Collecting lens ) – default
\lensDVG [Options] % Divergent (Scatter lens )
```

2.1 The Coordinates of the predefined Nodes

The following figure shows the coordinates of the predefined nodes (see Table 1).

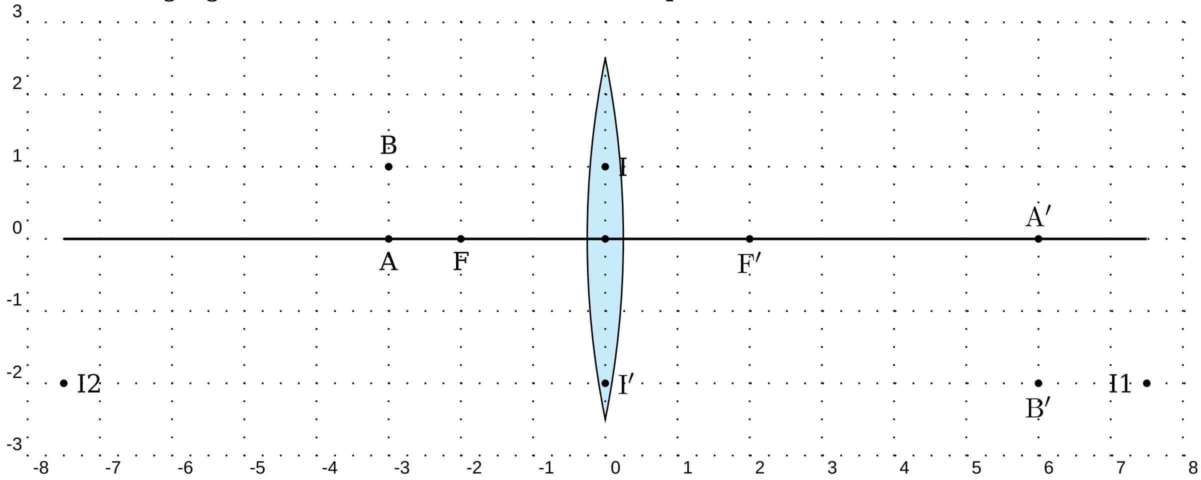
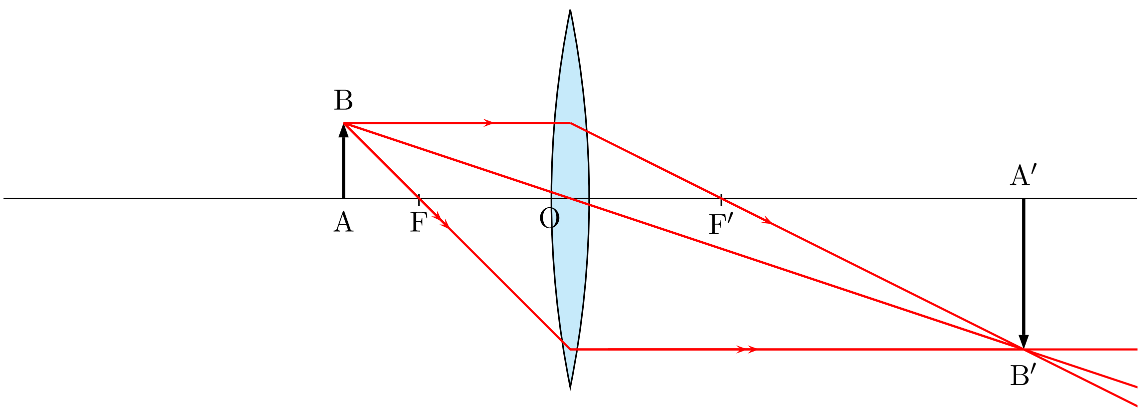


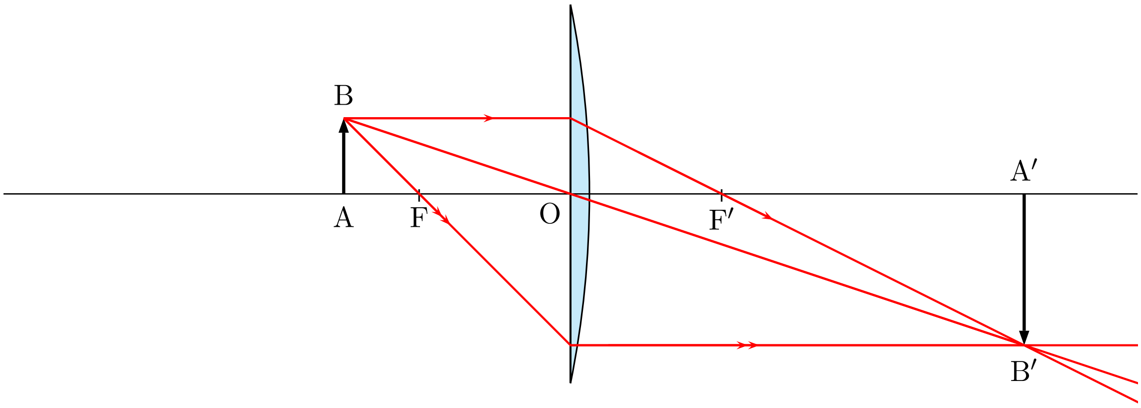
Figure 1: Coodinates of the predefined Nodes

2.2 The Lens Type



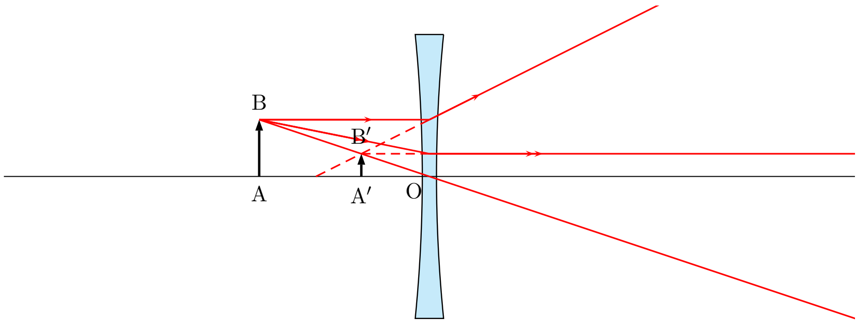

```
\lens[lensType=CVG]
```

Figure 2: Collecting lens



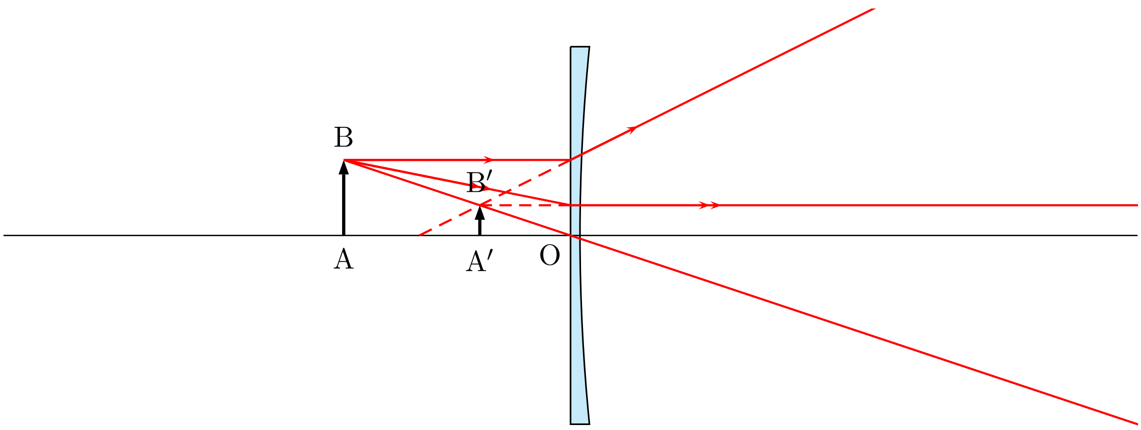

```
\lens[lensType=PCVG]
```

Figure 3: Plan Collecting lens




```
\psscalebox{0.75}{\lens[lensType=DVG,focus=-2,spotAi=270,spotBi=90]}
```

Figure 4: Scatter lens




```
\lens[lensType=PDVG,focus=-2,spotAi=270,spotBi=90]
```

Figure 5: Plan Scatter lens

Using \lens [lensType=...] gives the in figures 2 and 4 shown lenses with the default values from Table 2.

Table 2: Available options for lenses with the defaults

Option	*Name*	*Default*
Lense type (CVG|DVG|PCVG|PDVG)	lensType	CVG
Lense height in cm	lensHeight	5cm
Lense width in cm	lensWidth	0.5cm[1]
vertical scale (obsolet)	lensScale	1
View the lens	lensGlass	false
Second lens	lensTwo	false
Focus in cm	focus	2
Distance $\overline{\mathrm{OA}}$	OA	-4
Distance $\overline{\mathrm{AB}}$	AB	1.5
Lens color	lenscolor	
Arrow length in cm	lensarrowsize	0.2
Arrow inset in cm	lensarrowinset	0.5

[1] only for lensGlass=true , otherwise set to 2\pslinewidth

The origin of the coordinate system is by default vertically and horinzontally symmetric. If you want to place the lens at another coordinates then define your own pspicture-environment and use the \rput-command:

```
\begin{pspicture}(-7.5,-3)(7.5,3)
  \rput(x,y){\lens[...]}
\begin{pspicture}
\begin{pspicture*}(-7.5,-3)(7.5,3)
  \rput(x,y){\lens[...]}
\begin{pspicture*}
```

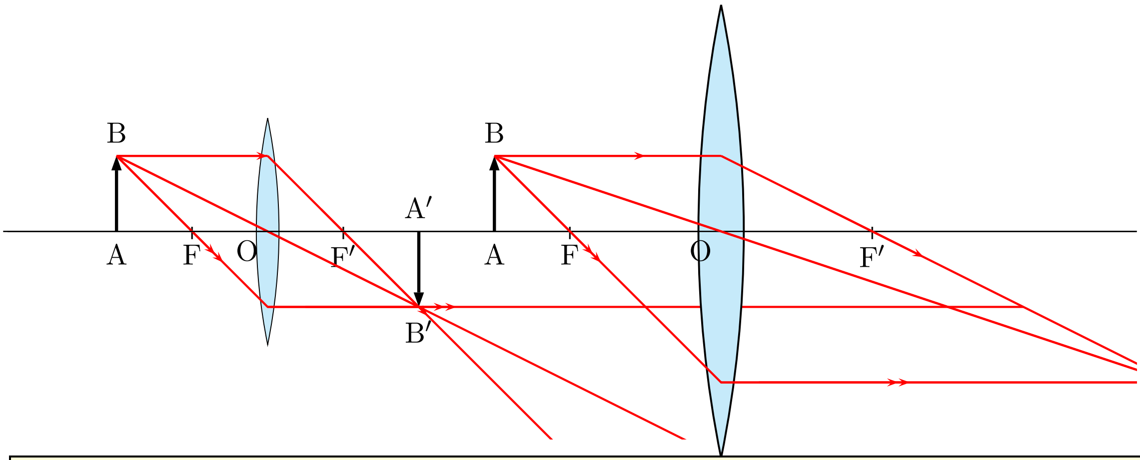
The star version enables the clipping option.

2.3 \Transform

The \Transform-macro renames all existing nodes in names with an additional „1“. Table 3 shows a list of all nodes. \Transform also defines a new node factice with the coordinates (X01,Y01). The renaming of all nodes makes it easier to handle objects with more than one lens. With the option lensTwo=true it is possible to chain the different rays of the lenses (Figure 8).

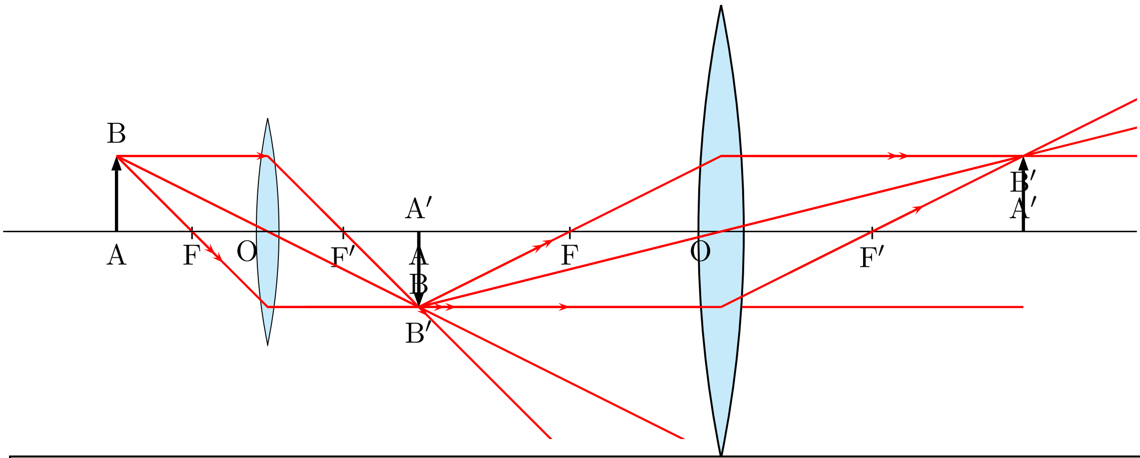
Table 3: Renaming of the nodes after calling the macro \Transform

old	A	B	A’	B’	O	F	F’	I	I’	XO	YO	OA’	A’B’
new	A1	B1	A’1	B’1	O1	F1	F’1	I1	I’1	XO1	YO1	O1A1’	A’1B’1



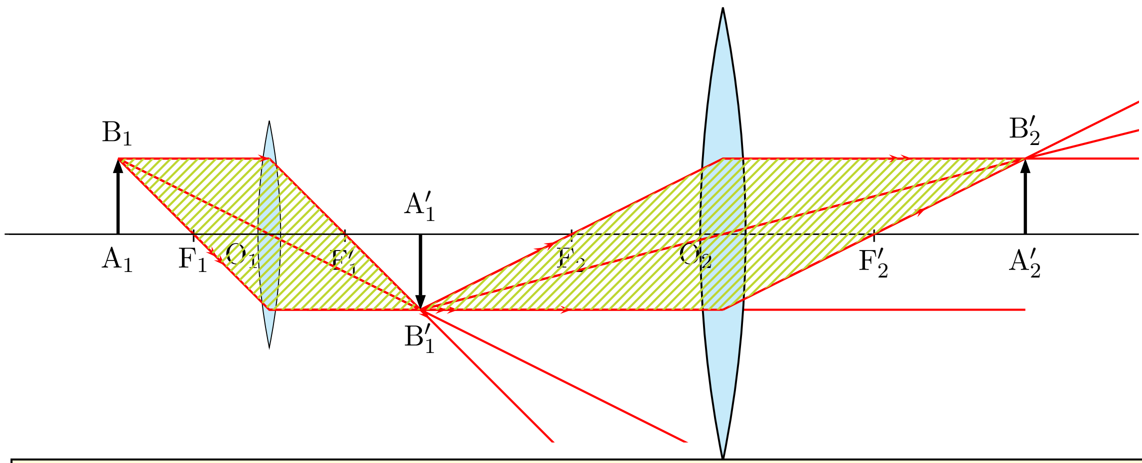

```
\begin{pspicture*}(-7.5,-2.75)(7.5,3)
\rput(0,0){\lens[lensScale=0.6,XO=-4,focus=1,OA=-2,lensGlass=true, lensWidth=0.5]}
\rput(0,0){\lens[lensScale=1.2,XO=2,focus=2,lensGlass=true,lensWidth=0.5]}
\end{pspicture*}
```

Figure 6: Definition of two unchained lenses




```
\begin{pspicture*}(-7.5,-2.75)(7.5,3)
\rput(0,0){\lens[lensScale=0.6,XO=-4,focus=1,OA=-2,lensGlass=, lensWidth=0.5]}
\Transform
\rput(0,0){\lens[lensScale=1.2,XO=2,focus=2,lensTwo=true,lensGlass=true,lensWidth
  =0.5]}
\end{pspicture*}
```

Figure 7: Definition of two chained lenses




```
\begin{pspicture*}(-7.5,-2.75)(7.5,3)
\rput(0,0){\lens[lensScale=0.6,XO=-4,nameF=F_1,nameA=A_1,nameB=B_1,
  nameFi=F’_1,nameAi={ },nameBi={},nameO=O_1,focus=1,OA=-2,lensGlass=true, lensWidth
    =0.5]}
\pspolygon[style=rayuresJaunes,linestyle=none](B)(I)(B’)(I’)(B)
\Transform
\rput(0,0){\lens[lensScale=1.2,XO=2,focus=2,nameA=A’_1,spotA=90,nameB=B’_1,spotB=270,
  nameO=O_2,nameAi=A’_2,spotAi=270,nameBi=B’_2,spotBi=90,nameF=F_2,nameFi=F’_2,
  lensTwo=true,lensGlass=true,lensWidth=0.5]}
\pspolygon[style=rayuresJaunes,linestyle=none](B)(I)(B’)(I’)(B)
\end{pspicture*}
```

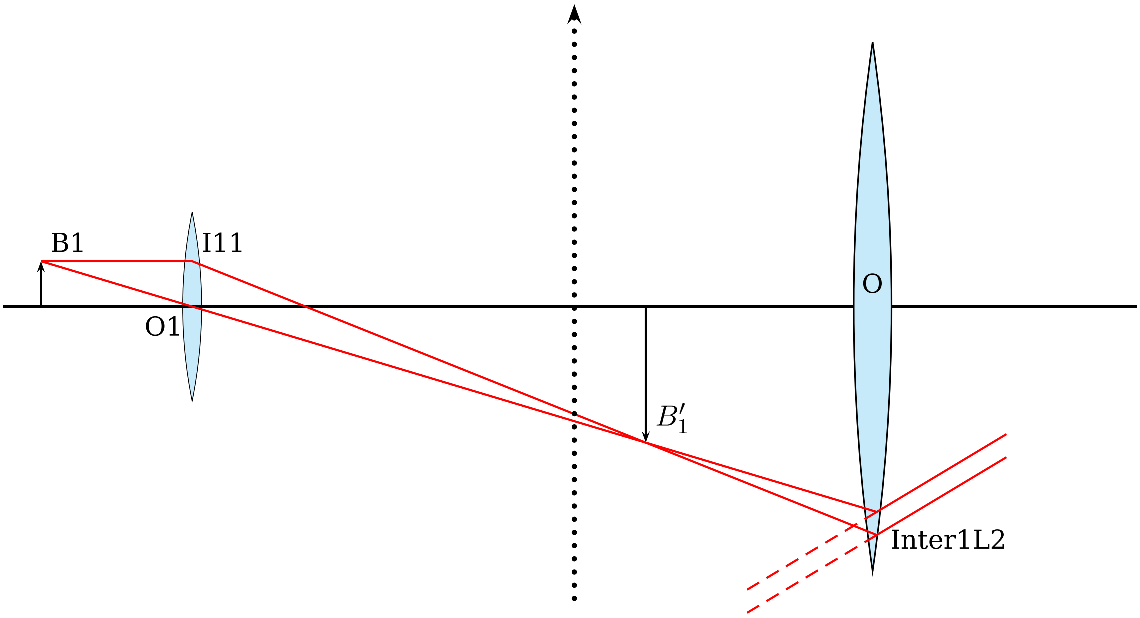
Figure 8: Definition of two chained lenses and an additional modification of the node labels.

2.4 \rayInterLens

This macro is only useful for a two-lens-system. Figure 9 shows such a system. The nodes `B1`, `I11`, `F'1`, and `B'1` are predefined by the `\lens`-macro. To draw the two rays from the left lense via the node `B'1` to the second lens, we need the coordinates of these points. `\rayInterLense` defines such nodes. The Syntax:

`\rayInterLense(`*StartNode*`)(`*IntermediatNode*`)(`*LensDistance*`){`*LensNode*`}`

Two parallel lines are drawn with the `\Parallel`-Macro.

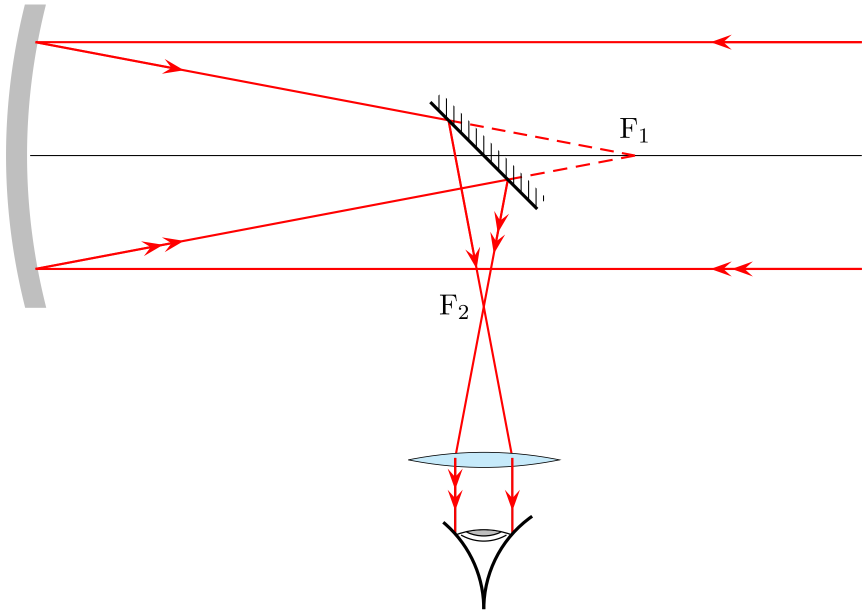



```
\begin{pspicture*}(-7.5,-4)(7.5,4)
\rput(0,0){\lens[focus=1.5,OA=-2,AB=0.6,XO=-5,lensGlass,yBottom=-4,yTop=4,drawing=false,
  lensWidth=0.5,lensScale=0.5,nameF=F_1,nameFi=F'_1]}
\psline[linewidth=1pt](xLeft)(xRight)
\Transform
\rput(0,0){\lens[focus=2,XO=4,lensGlass,yBottom=-4,yTop=4,drawing=false,lensWidth=0.5,
  lensHeight=7,nameF=F_2,nameFi=F'_2,spotF=90,spotFi=90]}
\psline{->}(A1)(B1)\psline{->}(A'1)(B'1)
\psset{linecolor=red}
\uput[45](B1){B1} \uput[90](O){O} \uput[225](O1){O1} \uput[45](I11){I11} \uput[45](B'1){$B'_1$}
\rayInterLens(I11)(B'1){4}{Inter1L2} \rayInterLens(O1)(B'1){4}{Inter2L2}
\uput[350](Inter1L2){Inter1L2}
\psline(B1)(I11)(B'1)(Inter1L2) \psline(B1)(O1)(B'1)(Inter2L2)
\Parallel(B'1)(O)(Inter2L2){B2inftyRigth} \Parallel(B'1)(O)(Inter1L2){B3inftyRigth}
\psset{length=-2,linestyle=dashed}
\Parallel(B'1)(O)(Inter2L2){B2inftyLeft} \Parallel(B'1)(O)(Inter1L2){B3inftyLeft}
\psline[linestyle=dotted, linewidth=2pt,linecolor=black]{->}(0,-4)(0,+4)
\end{pspicture*}
```

Figure 9: Demonstration of `\rayInterLens`

2.5 \telescope

Figure 10 shows the configuration of a telescope and Table 4 the special options for the \telescop-Macro.



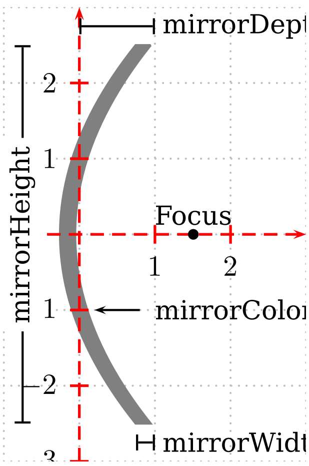

```
\telescope
```

Figure 10: \telescope-Macro

3 Mirrors

3.1 options

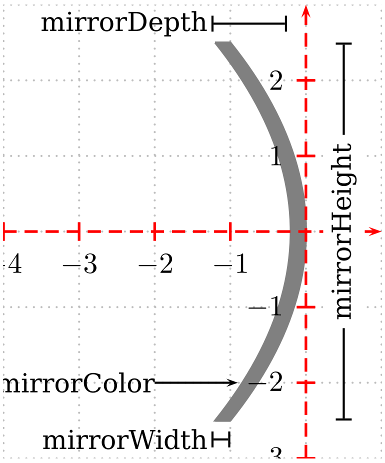
Figure 11 shows the available mirrors and Table 4 the possible options.



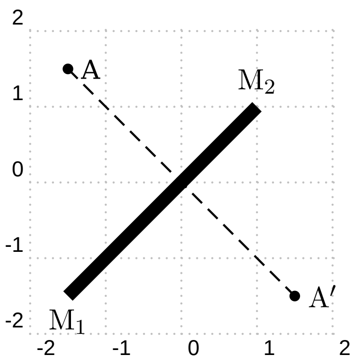

```
\begin{pspicture*}[showgrid=true](-1,-3)(3,3)
\rput(0,0){\mirrorCVG[mirrorColor=gray,drawing=false]}
\psaxes[linestyle=dashed,linecolor=red,linewidth=1pt,arrows=->](0,0)
   (-1,-3)(3,3)
\qdisk(Focus){2pt} \rput(Focus){\rput(0,0.25){Focus}}
\pcline[arrows=|-|](-0.75,-2.5)(-0.75,2.5)\ncput*[nrot=:U]{mirrorHeight}
\pcline[arrows=|-|](0,2.75)(1,2.75) \rput[l](1.1,2.75){mirrorDepth}
\pcline[arrows=|-|](1,-2.75)(0.75,-2.75) \rput[l](1.1,-2.75){mirrorWidth}
\rput[l](1,-1){mirrorColor}
\psline{<-}(0.2,-1)(0.8,-1)
\end{pspicture*}
```

Table 4: List of options for mirrors with the predefines values

Option	*Name*	*Default*
Left value of the picture in cm	`xLeft`	-0.5
Right value of the picture in cm	`xRight`	11
Lowest value of the picture in cm	`xBottom`	-6
Highest value of the picture in cm	`xTop`	2.5
Mirror height in cm	`mirrorHeight`	5
Mirror depth in cm	`mirrorDepth`	1
Mirror width in cm	`mirrorWidth`	0.25
Mirror color	`mirrorColor`	lightgray
Ray color	`rayColor`	black
Focus in cm (only together with the option `posMirrorTwo` senseful)	`mirrorFocus`	8
Position of the 2. mirror in cm	`posMirrorTwo`	8
Inclination of the 2. mirror in degrees	`mirrorTwoAngle`	45
Draw lines	`drawing`	true




```
\begin{pspicture*}[showgrid=true](-4,-3)(1,3)
\rput(0,0){\mirrorDVG[mirrorColor=gray,drawing=false]}
\psaxes[linestyle=dashed,linecolor=red,linewidth=1pt,arrows
  =->](0,0)(-4,-3)(1,3)
\qdisk(Focus){2pt}             \rput(Focus){\rput(0,0.25){Focus}}
\pcline[arrows=|-|](.5,-2.5)(.5,2.5) \ncput*[nrot=:U]{mirrorHeight}
\pcline[arrows=|-|](-1.25,2.75)(-.25,2.75) \rput[r](-1.3,2.75){
  mirrorDepth}
\pcline[arrows=|-|](-1.25,-2.75)(-1,-2.75) \rput[r](-1.3,-2.75){
  mirrorWidth}
\rput[r](-2,-2){mirrorColor}    \psline{->}(-2,-2)(-0.9,-2)
\end{pspicture*}
```

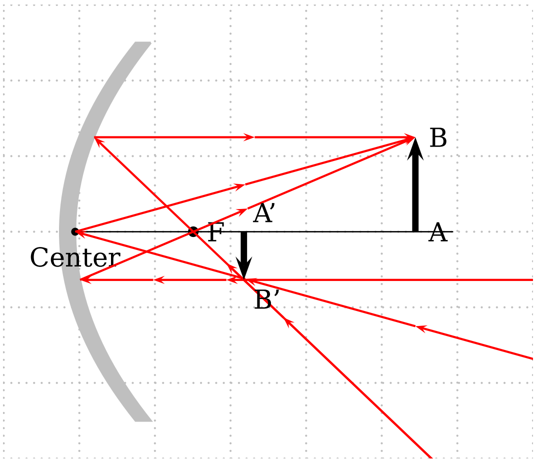



```
\begin{pspicture}[showgrid=true](-2,-2)(2,2)
\pnode(-1.5,-1.5){M1} \pnode(1,1){M2}
\uput[-90](M1){$\mathrm{M_1}$}\uput[90](M2){$\mathrm{M_2}$}
\pnode(-1.5,1.5){A}
\planMirrorRay(A)(M1)(M2){A'}
\psline[linewidth=5pt](M1)(M2)\pscircle*(A){2pt}
\uput[0](A){A} \uput[0](A'){$\mathrm{A'}$}
\pscircle*(A')(2pt} \psline[linestyle=dashed](A)(A')
\end{pspicture}
```

Figure 11: The different mirror macros: a) `\mirrorCVG` b) `\mirrorDVG` c) `\planMirrorRay`

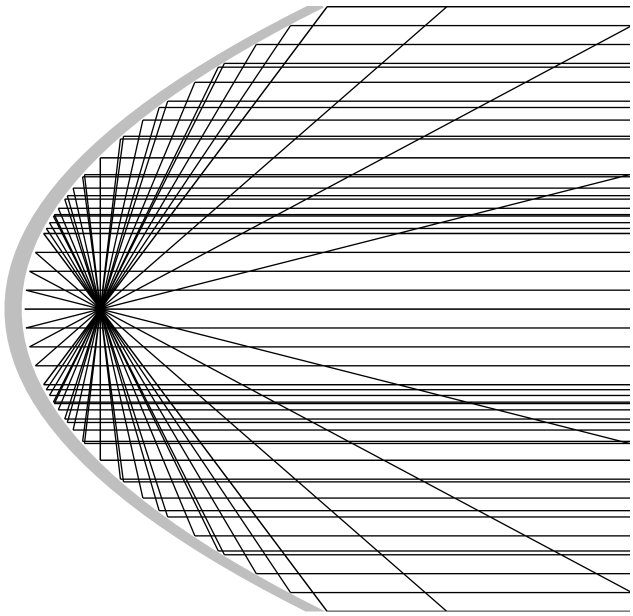
3.2 \mirrorCVG

Figure 12 shows the default for the `\mirrorCVG`-macro with the predefined nodes anf three default rays.




```
\begin{pspicture*}[showgrid=true](-1,-3)(6,3)
 \rput(0,0){\mirrorCVG[rayColor=red]}
\end{pspicture*}
```

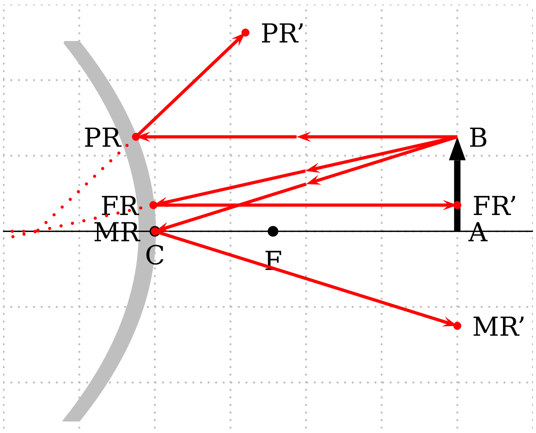
Figure 12: Parabolic Mirror `\mirrorCVG`



```
\begin{pspicture*}(-0.5,-4)(8,4)
 \rput(0,0){\mirrorCVG[mirrorHeight=8,
    mirrorDepth=4,drawing=false]}
 \multido{\rY=-4.00+0.25}{33}{%
   \mirrorCVGRay[linewidth=0.5pt,mirrorHeight
      =8,
     mirrorDepth=4](10,\rY)(1,\rY){Dummy}}
\end{pspicture*}
```

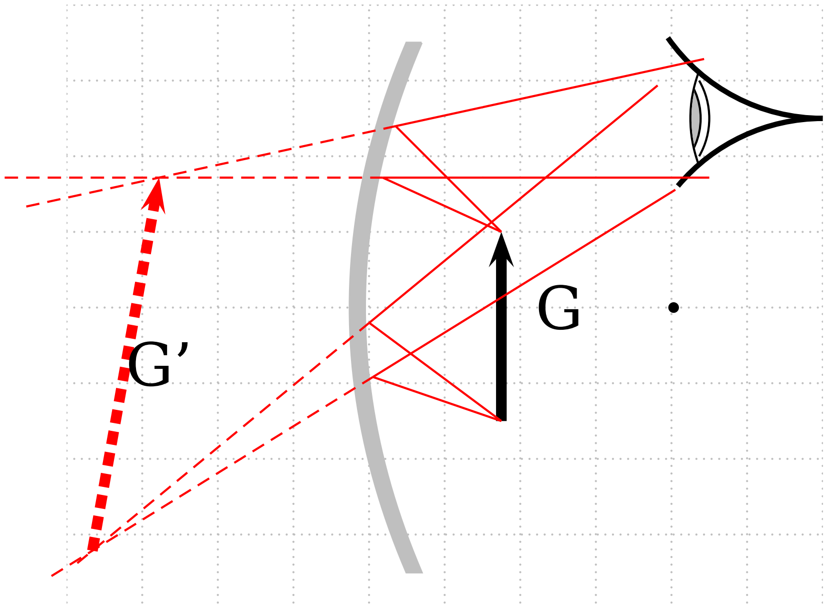
Figure 13: Example

4 \mirrorDVG




```
\begin{pspicture*}[showgrid=true](-2,-2.6)(5,3)
 \rput(0,0){\mirrorDVG[rayColor=red]}
\end{pspicture*}
```

Figure 14: \mirrorDVG




```
\begin{pspicture*}[showgrid=true](-4,-4)(6,4)
\rput(0,0){\mirrorCVG[drawing=false,mirrorDepth=0.75,mirrorHeight=7]}
\qdisk(Focus){2pt} \rput(6,2.5){\eye}
\pnode(1.75,-1.5){A}\pnode(1.75,1){B}\psline[arrows=->,linewidth=4pt](A)(B)
\uput{0.2}[0](2,0){\Huge G} \psset{linecolor=red}
\mirrorCVGRay[rayColor=red,mirrorHeight=7,mirrorDepth=0.75](A)(0,-0.9){P1}
\psOutLine[length=3](P1)(P1’){PEnd}\psBeforeLine[length=5,linestyle=dashed](P1)(P1’){PBegin}
\mirrorCVGRay[rayColor=red,mirrorHeight=7,mirrorDepth=0.75](A)(0,-0.2){P2}
\psOutLine[length=3](P2)(P2’’){PEnd}\psBeforeLine[length=5,linestyle=dashed](P2)(P2’){PBegin}
%
\mirrorCVGRay[rayColor=red,mirrorHeight=7,mirrorDepth=0.75](B)(0,2.75){P3}
\psOutLine[length=3](P3)(P3’){PEnd}\psBeforeLine[length=5,linestyle=dashed](P3)(P3’){PBegin}
\mirrorCVGRay[rayColor=red,mirrorHeight=7,mirrorDepth=0.75](B)(0,1.8){P4}
\psOutLine[length=3](P4)(P4’){PEnd}\psBeforeLine[length=5,linestyle=dashed](P4)(P4’){PBegin}
\ABinterCD(P3)(P3’)(P4)(P4’){A’}\ABinterCD(P1)(P1’)(P2)(P2’){B’}
\psline[arrows=->,linewidth=4pt,linestyle=dashed](B’)(A’)
\nodeBetween(A’)(B’){G’’}\uput{0}[0](G’’){\Huge G’}
\end{pspicture*}
```

Figure 15: Example as a magnifier

4.1 Drawing Rays in the Mirror Macros

There are two different macros for drawing rays:

```
\mirrorCVGRay [Options] (Node1)(Node2){MirrorNode}
\mirrorDVGRay [Options] (Node1)(Node2)(MirrorNode)
```

The `MirrorNode` maybe :

`MirrorNode`	first point on the mirror
`MirrorNode'`	end node or second point on the mirror if one more reflection happens
`MirrorNode"`	end node for a second reflection

If there are only one reflection, then `MirrorNode'` and `MirrorNode"` are the same.

4.2 `\planMirrorRay`

The `\planMirrorRay`-Macro caculates the coordinates of a mirrored point. In Figure 11 is a given node `A`, whereas `A'` is calculated by the macro. The syntax is:

```
\planMirrorRay(Mirrorbegin)(Mirrorend)(Originalpoint){New point}
```

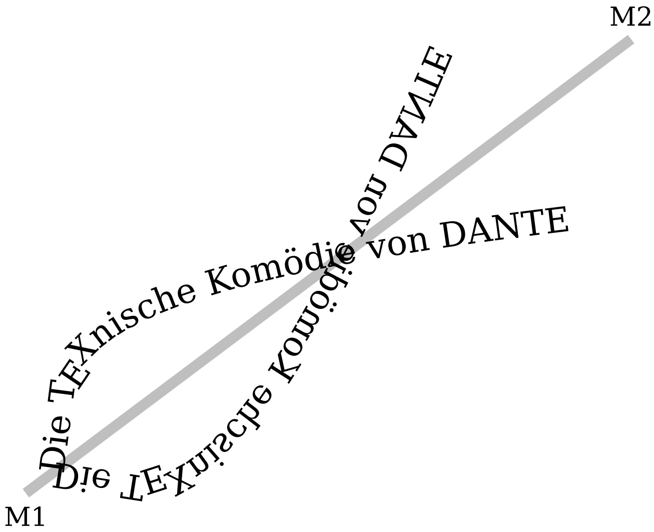
The macro doesn't draw any lines, only the coordinates of the new point are saved by the new node name.

4.3 `\symPlan`

`\symPlan` allows to mirroring complete plain graphical objects along a virtual center line. Figure 16 shows that this mirroring is a mathematical one and not a physical one. For more examples look at [**?**]. The syntax is:

```
\symPlan(node1)(node2){graphic object}
```

The two nodes define the mirror axis and the graphics object is in most cases a user defined macro, f.ex: This example needs the package `pst-text` for the `\pstextpath` macro.




```
\newcommand{\dtk}{\pstextpath(0,0){%
  \psplot[linestyle=none]{0}{8}{x sqrt sqrt 2 mul}}%
    {\Large Die \TeX{}nische Kom\"odie von DANTE}}
\begin{pspicture}(-4.5,-2.5)(2.5,5)
\pnode(-4,-2){M1} \uput[-90](M1){M1}
\pnode(4,4){M2}\uput[90](M2){M2}
\psline[linewidth=5\pslinewidth,linecolor=lightgray](M1)(M2)
\rput(-3.5,-1.75){\dtk}
\symPlan(M1)(M2){\rput(-3.5,-1.75){\dtk}}
\end{pspicture}
```

Figure 16: Demonstration of the \symPlan-Macro

4.4 Beam Light

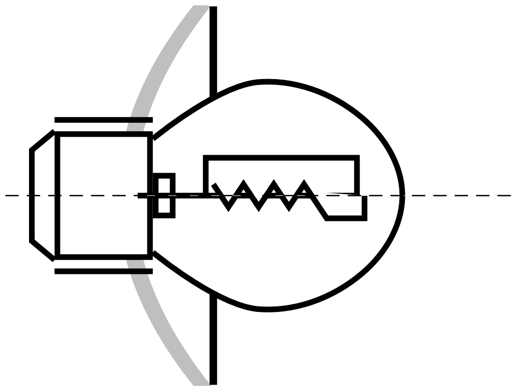
This macro is useful for the demonstration of high and low beam light. The syntax for this macro is:

```
\beamLight [Options]
```

The predefined options especially for the pspicture-coordinates are

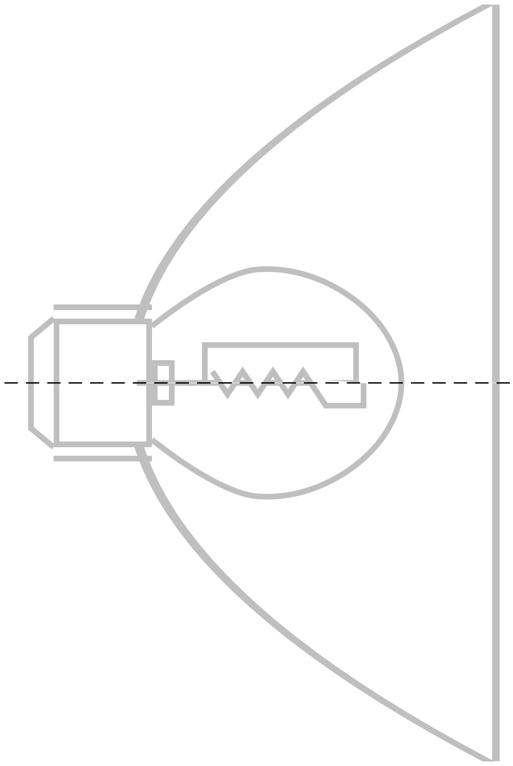
```
\psset[pst-optic]{xLeft=-5,xRight=5,yBottom=-5,yTop=5,drawing=false}% the default
```

You can place this macro with the \rput-command at any place in your own pspicture-environment.



```
\begin{pspicture}(-1,-3)(3,3)
  \rput(0,0){\beamLight}
\end{pspicture}
```

Figure 17: \beamLight without any Options



```
\begin{pspicture}(-1,-5.5)(5,5.5)
\rput(0,0){\beamLight[mirrorDepth=4.75,
  mirrorWidth=0.1,mirrorHeight=10,
  linecolor=lightgray]}
\end{pspicture}
```

Figure 18: \beamLight with Options

5 Refraction

6 \refractionRay

The syntax is

```
\refractionRay(A)(B)(C)(D){n1}{n2}{EndNode}
```

The macro uses the law of Snell

$$\frac{n_1}{n_2} = \frac{\sin\beta}{\sin\alpha} \tag{1}$$

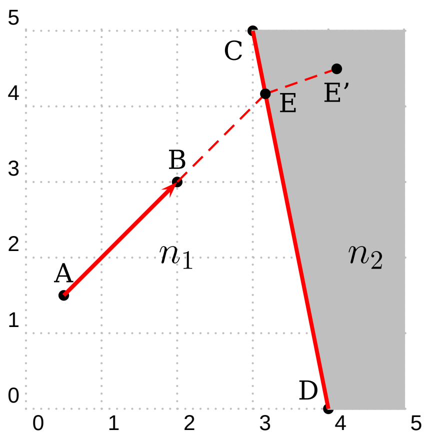
where the n_1 and n_2 are the refraction numbers with the predefined values

$$n_1 = 1 \tag{2}$$
$$n_2 = 1.41 \tag{3}$$

and α the incoming abd β the outgoing angle of the ray.

The refractionnumbers have the internal names `refractA` and `refractB`.

A total reflection instead of a refraction is possible, when the ray starts in a medium with a higher refrectionnumber. This happens when $\sin\beta > 1$ in Equ. 1. In this case we have $\alpha = \beta$, a total reflection.




```
\begin{pspicture}[showgrid=true](5,5)
\pnode(0.5,1.5){A}\qdisk(A){2pt}\uput[90](A){A}
\pnode(2,3){B}\qdisk(B){2pt}\uput[90](B){B}
\psline[linewidth=1.5pt,linecolor=red]{->}(A)(B)
\pnode(3,5){C}\qdisk(C){2pt}\uput[225](C){C}
\pnode(4,0){D}\qdisk(D){2pt}\uput[135](D){D}
\pspolygon[fillstyle=solid,fillcolor=lightgray,
  linecolor=lightgray](C)(D)(5,0)(5,5)(C)
\psline[linewidth=1.5pt,linecolor=red](C)(D)
\rput(2,2){\Large$n_1$} \rput(4.5,2){\Large$n_2$}
\refractionRay(A)(B)(C)(D){1}{4}{E}
\psline[linestyle=dashed,linecolor=red](B)(E)
\psline[linestyle=dashed,linecolor=red](E)(E')
\qdisk(E){2pt}\uput[-20](E){E}
\qdisk(E'){2pt}\uput[-90](E'){E'}
\end{pspicture}
```

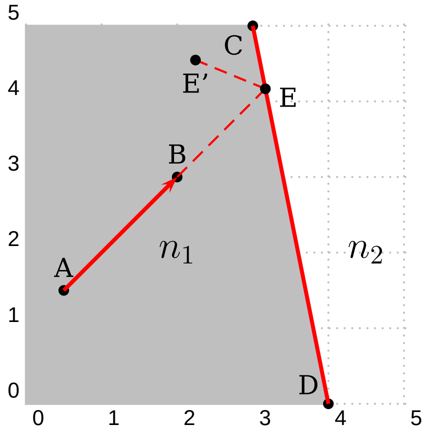
The macro needs the values for the four nodes, the two refractionnumbers and the name for the end node. As you can see in the figure the end node of the ray is the intermediate point between the linear ray and the linear medium. The end node of the refracted ray has the same name with an additional single quotation mark. In the figure the macro was called as

```
\refractionRay(A)(B)(C)(D){1}{4}{E}
```

$$n_1 < n_2 \tag{4}$$

It is no problem to draw a ray which is going straight through another medium. It can be done by using the macro twice as shown in the following examples.

6.1 Total Reflection




```
\begin{pspicture}[showgrid=true](5,5)
\pnode(0.5,1.5){A}\pnode(2,3){B}
\pnode(3,5){C} \pnode(4,0){D}
\pspolygon[fillstyle=solid,fillcolor=lightgray,
  linecolor=lightgray](C)(D)(0,0)(0,5)(C)
\qdisk(A){2pt}\uput[90](A){A}
\qdisk(B){2pt}\uput[90](B){B}
\qdisk(C){2pt}\uput[225](C){C}
\qdisk(D){2pt}\uput[135](D){D}
\psline[linewidth=1.5pt,linecolor=red]{->}(A)(B)
\psline[linewidth=1.5pt,linecolor=red](C)(D)
\rput(2,2){\Large$n_1$}\rput(4.5,2){\Large$n_2$}
\refractionRay(A)(B)(C)(D){4}{1}{E}
\psline[linestyle=dashed,linecolor=red](B)(E)
\psline[linestyle=dashed,linecolor=red](E)(E')
\qdisk(E){2pt}\uput[-20](E){E}
\qdisk(E'){2pt}\uput[-90](E'){E'}
\end{pspicture}
```

In the figure the macro was called as

```
\refractionRay(A)(B)(C)(D){4}{1}{E}
```

$$n_1 > n_2 \tag{5}$$

7 Prism

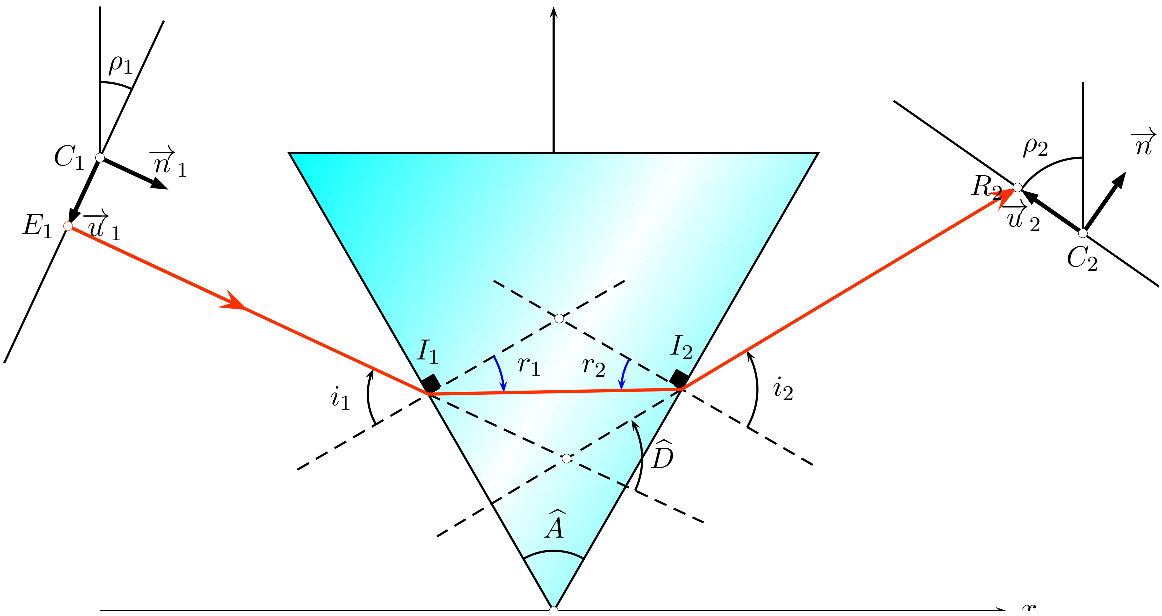
This command allows to simulate the deviation of a mono-chromatic light ray with a prism. There are only few parameters. The indicated values are the default ones.

name	*meaning*	*default*
`AnglePrism`	The angle to the top of prism.	60
`AnglePlan1`	The angle between the plane (1), where the tranmitter takes place, and the vertical. Negative values are allowed.	25
`AnglePlan2`	The angle between the plane (2) (the screen), and the vertical. Negative values are allowed.	55
`k`	Position of transmitter relative to choosen origin C_1 on the plane : $\overrightarrow{C_1E_1} = k\overrightarrow{u_1}$.	1
`lambda`	The wavelength , in nm.	632.8
`notations`	The plane where transmitting source takes place, with all indications, origin, angle, etc., as well as the screen are displayed by default. This can be useful in order to finalize a figure, but it is possible to deactivate this feature with the option.	`false`

With `AnglePlan1` the incident ray direction can be changed. The incidence spot changes according to `k`.

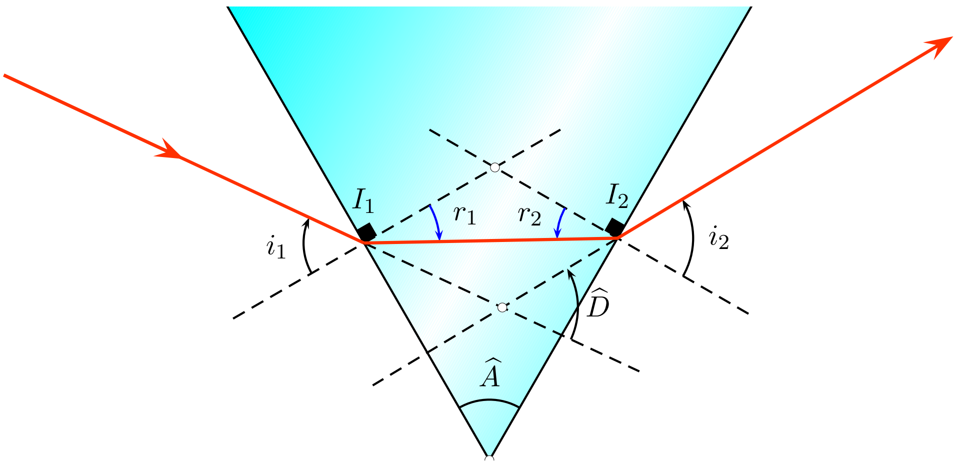
The outline of processing we have adopted is the Gernot Hoffmann one. For more details look into the document: http://www.fho-emden.de/~hoffmann/prism16072005.pdf

7.1 Figure with default values and construction indications



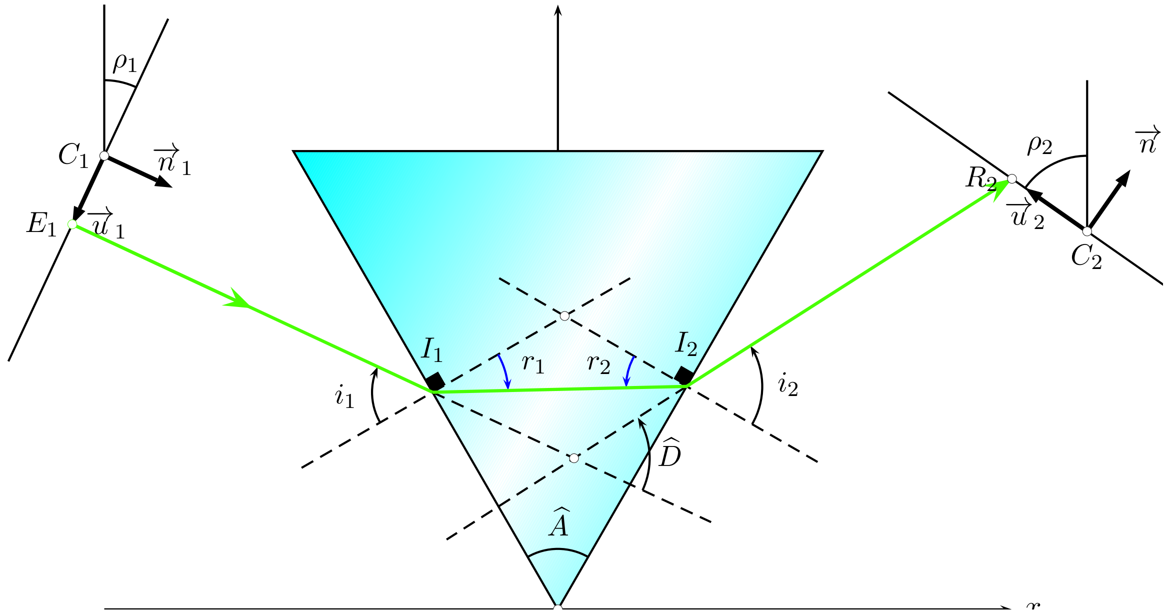

```
\begin{pspicture*}(-8,0)(8,8)
\psprism
\end{pspicture*}
```

7.2 Figure with default values, without construction indications




```
\begin{pspicture*}(-8,0)(8,6)
\psprism[notations=false]
\end{pspicture*}
```

7.3 Color matches wavelength



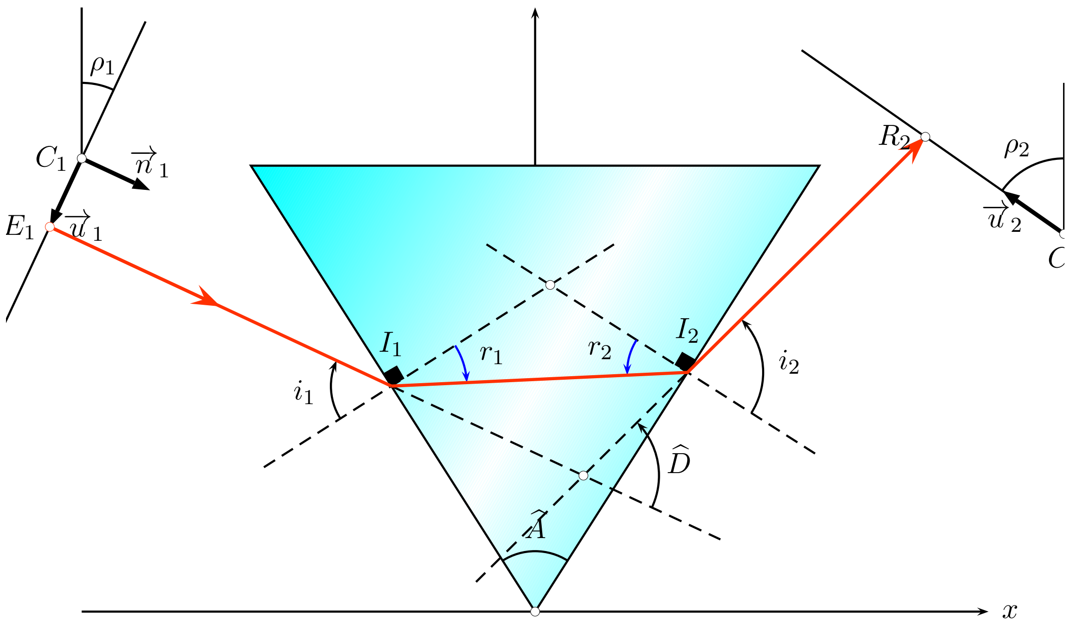

```
\begin{pspicture*}(-8,0)(8,8)
\psprism[lambda=530]%
\end{pspicture*}
```

Note: we have not planned physical impossibilities. When r_2 is greater than the limit angle, there is no transmission in air, and it's impossible to calculate i_2. Then, we have a PostScript message:

```
Displaying page 1
Displaying page 2
Displaying page 3
Displaying page 4
Error: /rangecheck in --sqrt--
Operand stack:
  alpha2 -1.02701 -0.0547467
```

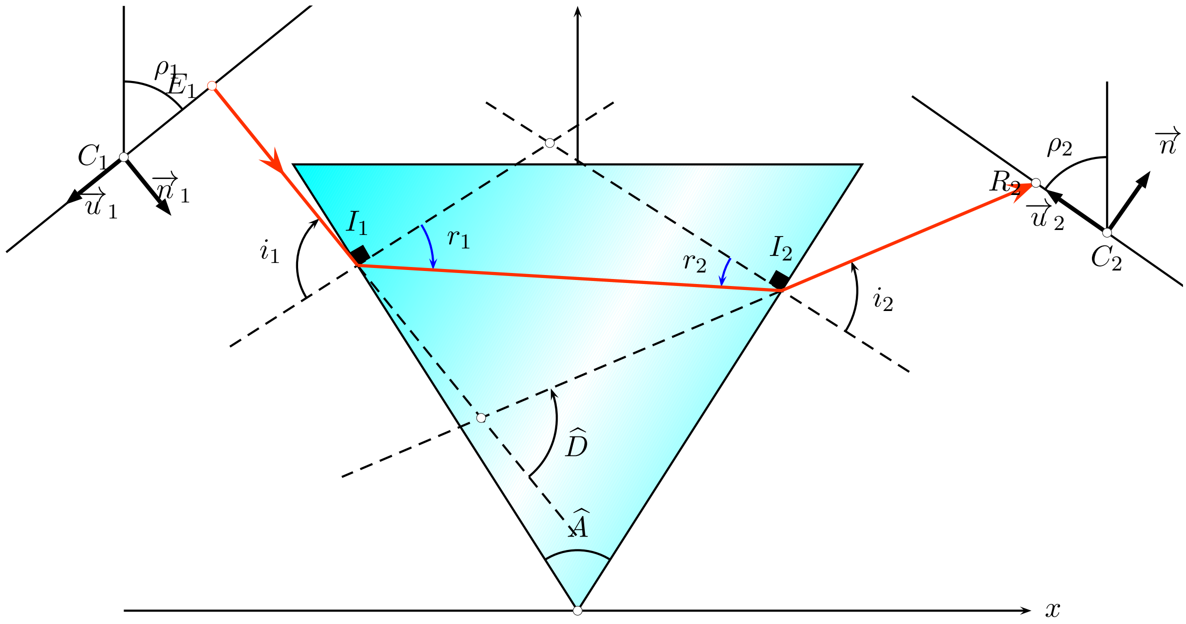
We remind you that `alpha2` is i_2.

For instance, `AnglePrism=65`, other default parameters remains unchanged.




```
\begin{pspicture*}(-7,-0.2)(7,8)
\psprism[AnglePrism=65]
\end{pspicture*}
```

It will become right when we change the incident ray slope:




```
\begin{pspicture*}(-8,-0.2)(8,8)
\psprism[AnglePrism=65,AnglePlan1=51,k=-1.5]
\end{pspicture*}
```

We choose k=-1.5 in order to have a incident ray which strikes (?) the input side roudly in its center. But, in these particular cases, the physicist know-how is important (*bis repetita*). Isn't it?

8 Spherical Optic

8.1 \lensSPH

The syntax is

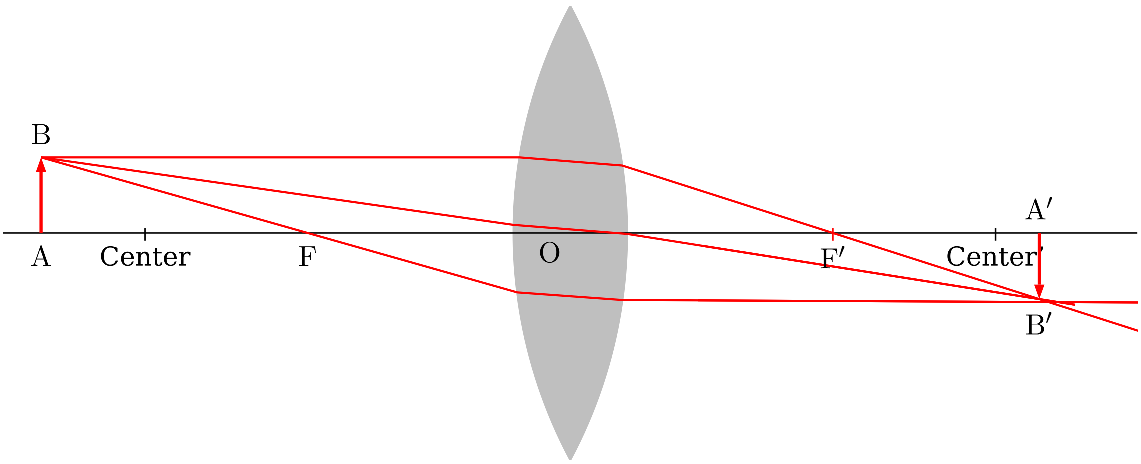
```
\lensSPH [Options]
```

It changes some default values for the options to:

meaning	name	default
Object Distance in cm	`OA`	-7
Lens Height in cm	`lensHeight`	6
Lens Width in cm	`lensWidth`	1.5
Refraction Number n_2	`refractB`	2

Convergent Lens

Without any option it draws a spherical convergent lens. `\lensSPH` is equivilant to `\lensSPH[`*lensType=CVG* `]`.



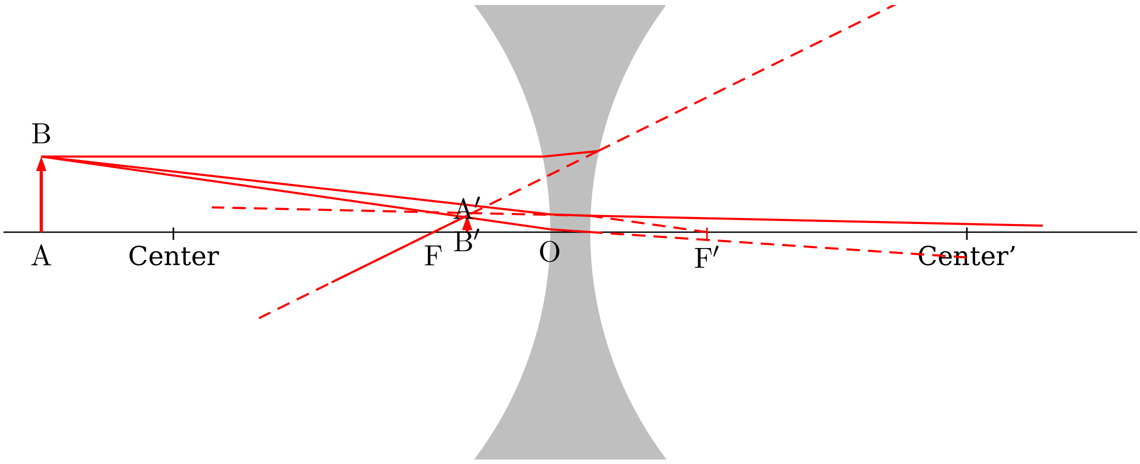

```
\lensSPH
```

Divergent Lens

The syntax is

```
\lensSPH [lensType=DVG ,...]
```

It draws a spherical divergent lens:




```
\lensSPH[lensType=DVG,lensWidth=0.5]
```

It changes some default values for the options in the same way as for the convergent lens.

8.2 Options

The macro uses the law of Snell

$$\frac{n_1}{n_2} = \frac{\sin\beta}{\sin\alpha} \tag{6}$$

where the n_1 and n_2 are the refraction numbers with the predefined values

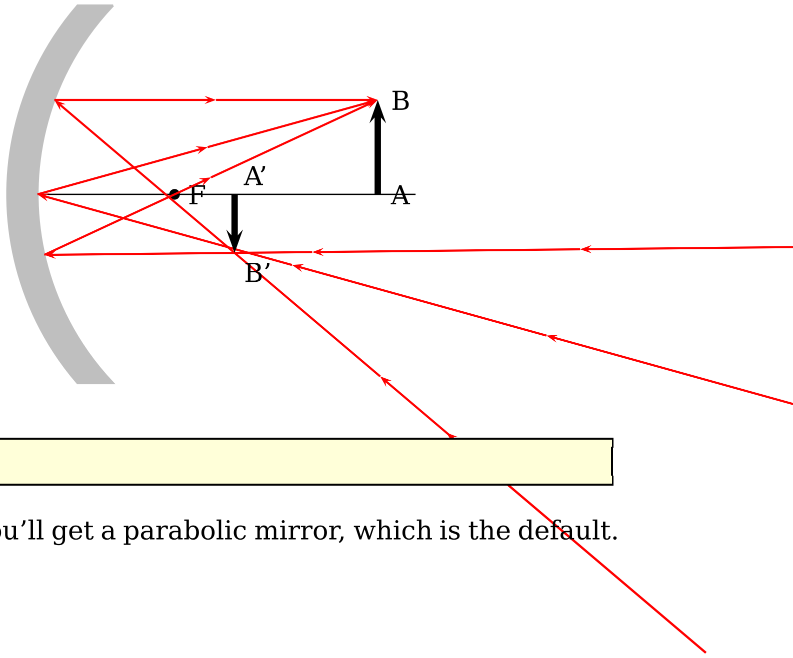
$$n_1 = 1 \tag{7}$$
$$n_2 = 1.41 \tag{8}$$

and α the incoming abd β the outgoing angle of the ray.
The refractionnumbers have the internal names `refractA` and `refractB`.

9 \mirrorCVG

The syntax is

```
\mirrorCVG [mirrorType=SPH ]
```



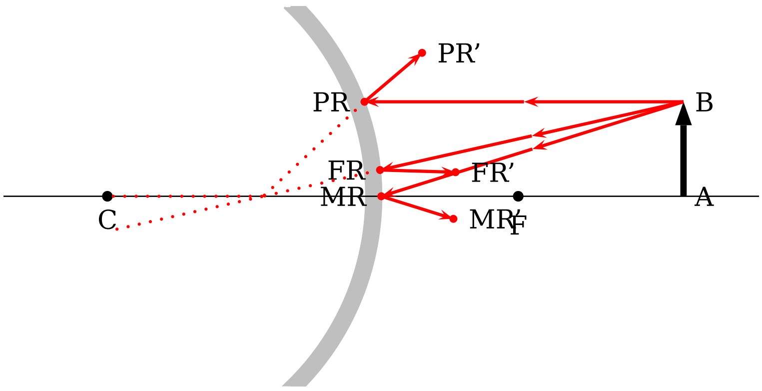

```
\mirrorCVG[mirrorType=SPH]
```

Without the option `mirrorType=SPH` you'll get a parabolic mirror, which is the default.

10 \mirrorDVG

The syntax is

```
\mirrorDVG [mirrorType=SPH ]
```

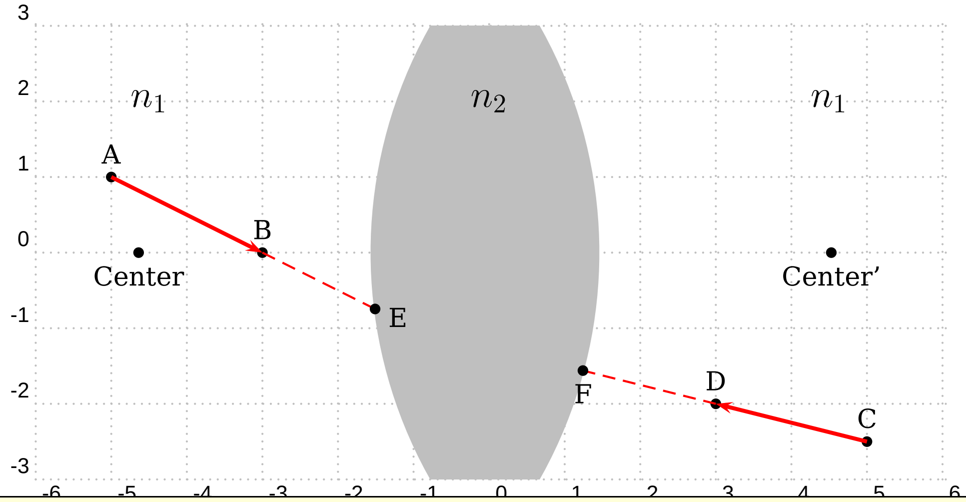



```
\mirrorDVG[mirrorType=SPH]
```

Without the option `mirrorType=SPH` you'll get a parabolic mirror (option PARA).

11 \ABinterSPHLens

The syntax is




```
\begin{pspicture}[showgrid=true](-6,-3)(6,3)
\rput(0,0){\lensSPH[lensType=CVG,lensHeight=8,lensWidth=3,drawing=false]}
\qdisk(Center){2pt}\uput[-90](Center){Center}
\qdisk(Center')(2pt}\uput[-90](Center'){Center'}
\pnode(-5,1){A}\qdisk(A){2pt}\uput[90](A){A}
\pnode(-3,0){B}\qdisk(B){2pt}\uput[90](B){B}
\psline[linewidth=1.5pt,linecolor=red]{->}(A)(B)
\pnode(5,-2.5){C}\qdisk(C){2pt}\uput[90](C){C}
\pnode(3,-2){D}\qdisk(D){2pt}\uput[90](D){D}
\psline[linewidth=1.5pt,linecolor=red]{->}(C)(D)
\rput(-4.5,2){\Large $n_1$}\rput(0,2){\Large $n_2$}\rput(4.5,2){\Large $n_1$}
\ABinterSPHLens(A)(B)(Center'){E}\ABinterSPHLens(C)(D)(Center){F}
\psline[linestyle=dashed,linecolor=red](B)(E)
\psline[linestyle=dashed,linecolor=red](D)(F)
\qdisk(E){2pt}\uput[-20](E){E}\qdisk(F){2pt}\uput[-90](F){F}
\end{pspicture}
```

The macro needs two nodes for the rays, the coordinates/nodes of the center/middle of the sperical lens and a name of the intermediate node.In the figure the macro was called as

```
\ABinterSPHLens(A)(B)(Center'){E}
\ABinterSPHLens(C)(D)(Center){F}
```

12 \lensSPHRay

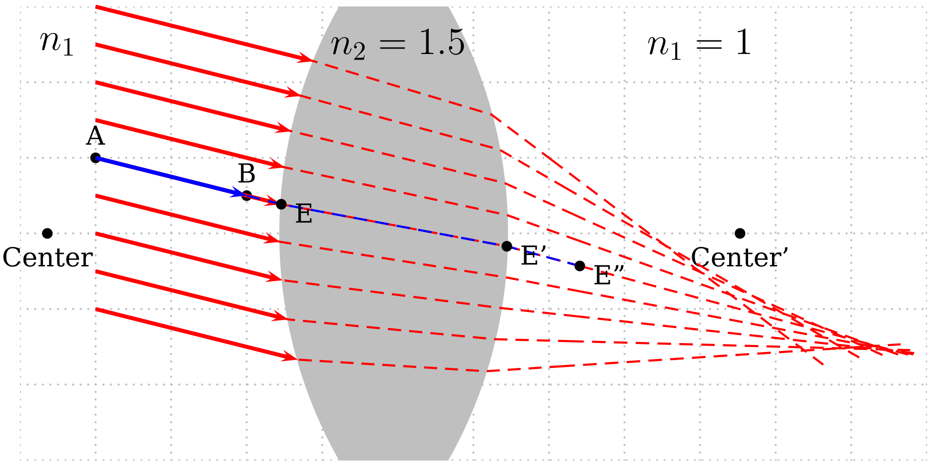
The syntax is

\lensSPHRay [Options] (*A*)(*B*){*refractA*}{*refractB*}{*NodeName*}

This macro calculates the coordinates of the given ray $\overline{\mathrm{AB}}$ on its way into the lens. The only possible option `rightRay=false|true`[2] enables rays from the right to the left.

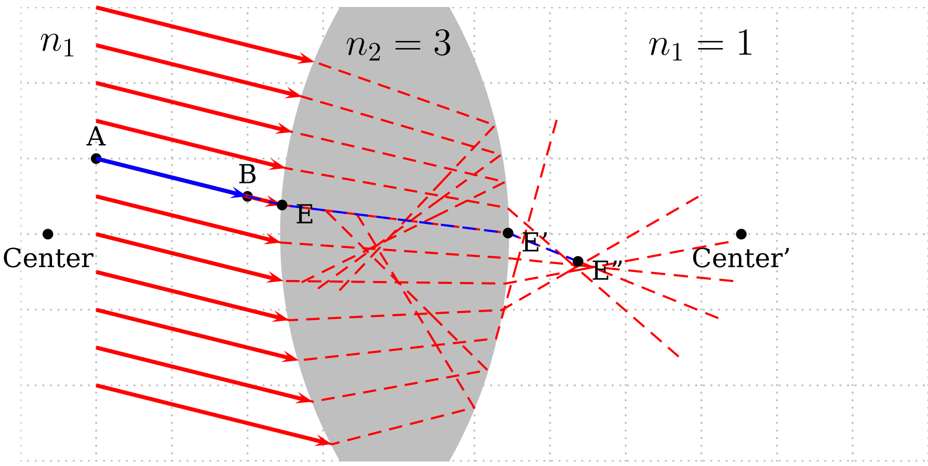
2 Default is `false`

There are still some problems with this option but try it out.




```
\begin{pspicture*}[showgrid=true](-5,-3)(7,3)
\rput(0,0){\lensSPH[lensType=CVG,lensHeight=8,lensWidth=3,drawing=false]}
\qdisk(Center){2pt}\uput[-90](Center){Center}
\qdisk(Center')(2pt}\uput[-90](Center'){Center'}
\pnode(-4,1){A}\qdisk(A){2pt}\uput[90](A){A}
\pnode(-2,0.5){B}\qdisk(B){2pt}\uput[90](B){B}
\rput(-4.5,2.5){\Large $n_1$}\rput(0,2.5){\Large $n_2=1.5$}\rput(4,2.5){\Large $n_1=1$}
\multido{\rA=3+-0.5,\rB=2.5+-0.5}{9}{%
  \lensSPHRay[rightRay=false](-4,\rA)(-2,\rB){1}{1.5}{F}
  \psline[linewidth=1.5pt,linecolor=red]{->}(-4,\rA)(F)
  \psline[linestyle=dashed,linecolor=red](-4,\rA)(F)(F')(F'')
  \psOutLine[linestyle=dashed,linecolor=red,length=4.5](F')(F''){FEnd}}
\psline[linewidth=1.5pt,linecolor=blue]{->}(A)(B)
\lensSPHRay[lensType=CVG](A)(B){1}{1.5}{E}
\psline[linestyle=dashed,linecolor=blue](B)(E)(E')(E'')
\qdisk(E){2pt}\uput[-20](E){E}\qdisk(E'){2pt}\uput[-20](E'){E'}
\qdisk(E''){2pt}\uput[-20](E''){E''}
\end{pspicture*}
```

And the same with $n_2 = 3$:



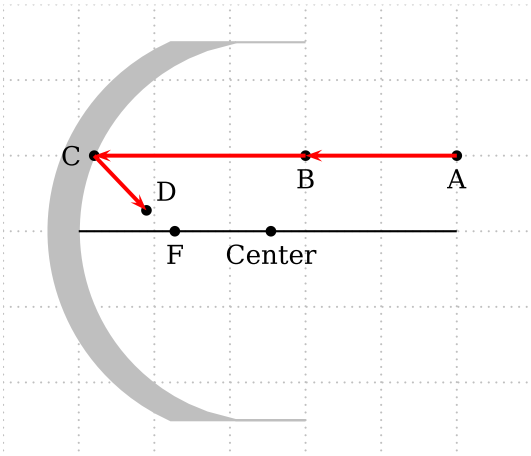

```
\begin{pspicture*}[showgrid=true](-5,-3)(7,3)
\rput(0,0){\lensSPH[lensType=CVG,lensHeight=8,lensWidth=3,drawing=false]}
\qdisk(Center){2pt}\uput[-90](Center){Center}
\qdisk(Center')\{2pt}\uput[-90](Center'){Center'}
\pnode(-4,1){A}\qdisk(A){2pt}\uput[90](A){A}
\pnode(-2,0.5){B}\qdisk(B){2pt}\uput[90](B){B}
\rput(-4.5,2.5){\Large $n_1$}\rput(0,2.5){\Large $n_2=3$}\rput(4,2.5){\Large $n_1=1$}
\multido{\rA=3+-0.5,\rB=2.5+-0.5}{11}{%
 \lensSPHRay[rightRay=false](-4,\rA)(-2,\rB){1}{3}{F}
 \psline[linewidth=1.5pt,linecolor=red]{->}(-4,\rA)(F)
 \psline[linestyle=dashed,linecolor=red](-4,\rA)(F)(F')(F'')
 \psOutLine[linestyle=dashed,linecolor=red](F')(F''){FEnd}}
\psline[linewidth=1.5pt,linecolor=blue]{->}(A)(B)
\lensSPHRay[lensType=CVG](A)(B){1}{3}{E}
\psline[linestyle=dashed,linecolor=blue](B)(E)(E')(E'')
\qdisk(E){2pt}\uput[-20](E){E}\qdisk(E'){2pt}\uput[-20](E'){E'}
\qdisk(E''){2pt}\uput[-20](E''){E''}
\end{pspicture*}
```

13 \reflectionRay

The syntax is

`\reflectionRay [Options] (`*A*`)(`*B*`){`*NodeName*`}`

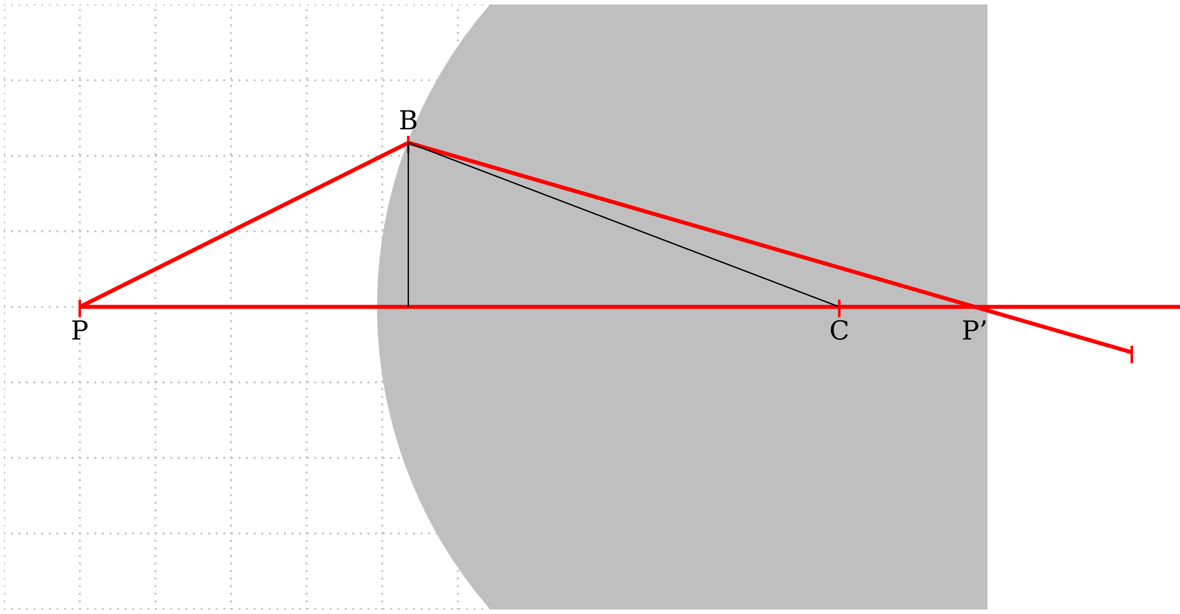
This macro calculates the coordinates of the given ray $\overline{\mathrm{AB}}$ on its way out of the mirror. The only senseful option is `mirrorType=CVG` or `mirrorType=DVG`. The most important fact is that the point `B` must be the one on the mirror. If you do not know it's coordinates you can use the macro `ABinterSPHLens[lensType=CVG](A1)(A2)(Center){NodeName)`, which calculates the coordinates of the intermediate point.




```
\begin{pspicture*}[showgrid=true](-1,-3)(6,3)
\rput(0,0){%
  \mirrorCVG[mirrorType=SPH,mirrorHeight=5,mirrorWidth=0.2,yBottom=-3,yTop=3,drawing=false,
    mirrorDepth=3]
  \qdisk(Center){2pt}\qdisk(Focus){2pt}\uput[-90](Center){Center}\uput[-90](Focus){F}
  \psline(O)(xRight)}
\ABinterSPHLens(5,1)(3,1)(Center){C}
\reflectionRay[mirrorType=CVG-SPH](5,1)(C){D}
\qdisk(5,1){2pt}\uput[-90](5,1){A}\qdisk(3,1){2pt}\uput[-90](3,1){B}
\qdisk(C){2pt}\uput[180](C){C}\qdisk(D){2pt}\uput[45](D){D}
\psset{linewidth=1.5pt,linecolor=red,arrows=->}
\psline(5,1)(3,1)\psline(3,1)(C)\psline(C)(D)
\end{pspicture*}
```

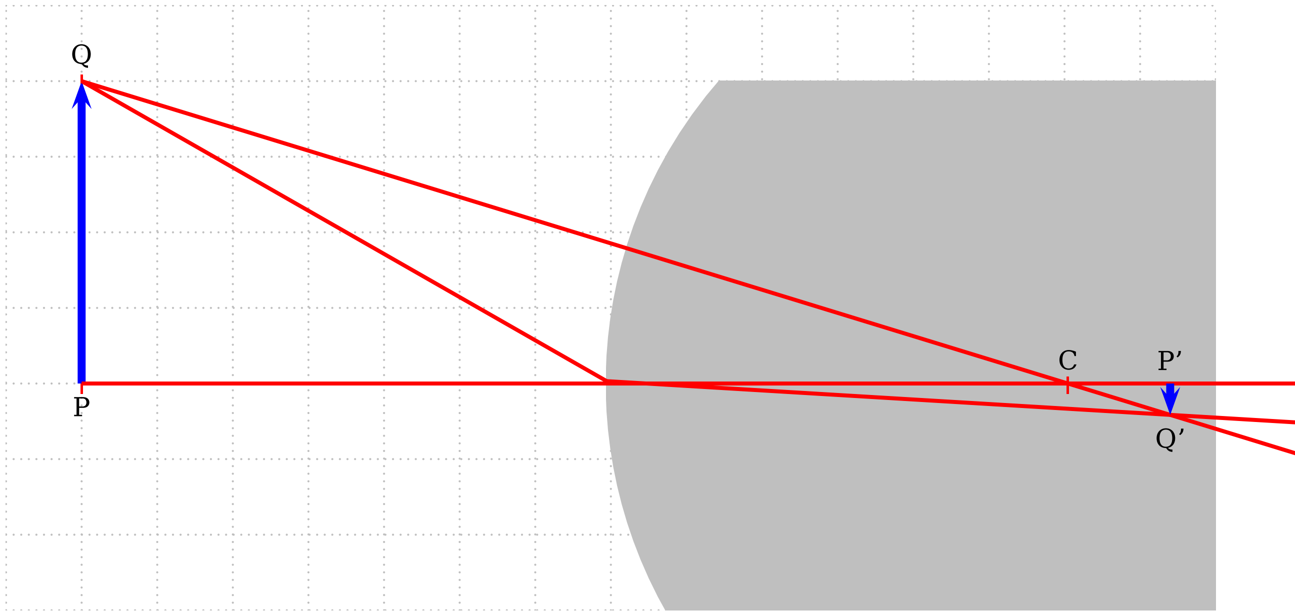
13.1 Refraction at a Spherical surface

Construction for finding the position of the image point P' of a point object P formed by refraction at a sperical surface.




```
\begin{pspicture*}[showgrid=true](-10,-4)(3,4)
\rput(0,0){%
 \lensSPH[lensType=CVG,lensHeight=12,lensWidth=10,yBottom=-6,yTop=6,xLeft=-6,xRight=6,drawing=
    false]}
\psset{linecolor=red,linewidth=1.5pt,dotstyle=|}
\pnode(-9,0){P}\psdots(P)\uput[-90](P){P}\psline(P)(xRight)
\lensSPHRay(P)(-5,2){1}{9}{Q} \psline(P)(Q)(Q') \psdots(Q)\uput[90](Q){B}
\ABinterCD(Q)(Q')(0,0)(5,0){P'}\psdots(Q')\uput[-90](P'){P'}
\psline[linewidth=0.5pt,linecolor=black](Center')(Q)\psline[linewidth=0.5pt,linecolor=black](Q)(Q
  |0,0)
\psdots(Center')\uput[-90](Center'){C}
\end{pspicture*}
```

Construction for determining the height of an image formed by refraction at a sperical surface.




```
\begin{pspicture*}[showgrid=true](-13,-3)(3,5)
\rput(0,0){%
  \lensSPH[lensType=CVG,lensHeight=12,lensWidth=10,yBottom=-4,yTop=4,xLeft=-5,xRight=5,drawing=
    false]}
\psset{linecolor=red,linewidth=1.5pt,dotstyle=|}
\pnode(-12,0){P}\psdots(P)\uput[-90](P){P}\pnode(-12,4){Q}\psdots(Q)\uput[90](Q){Q}
\psline[linecolor=blue,linewidth=3pt,arrows=->](P)(Q)\psline(P)(xRight)
\lensSPHRay(Q)(Center'){1}{9}{S1}\lensSPHRay(Q)(-5,0){1}{9}{S2}%
\psline(Q)(S1')\psline(Q)(S2)(S2')\ABinterCD(Q)(S1')(S2)(S2'){Q'}\pnode(Q'|0,0){P'}
\psline[linecolor=blue,linewidth=3pt,arrows=->](P')(Q')
\uput[90](P'){P'}\uput[-90](Q'){Q'}\psdots(Center')\uput[90](Center'){C}
\end{pspicture*}
```

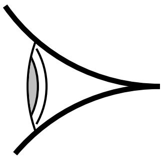
14 Utility Macros

14.1 \eye

Syntax:

```
\eye
```

There are no Options for this symbol of an human eye (Figure **??**). Use the `\rput`-macro to put the eye elsewhere.



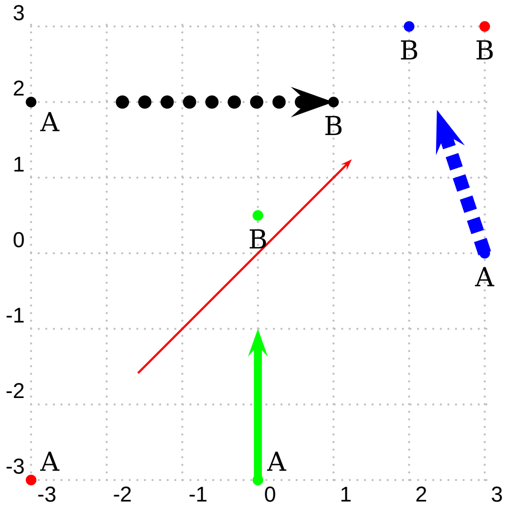
```
\begin{pspicture}(-1,-0.75)(1,0.75)
  \rput(1,0){\eye}
\end{pspicture}
```

15 \Arrows

Syntax wirh the following options:

`\Arrows [Options] (`*NodeA*`)(`*NodeB*`)`

Option	*Name*	*Standard*
Offset for arrow start in cm	`posStart`	0
Length of the arrow in cm	`length`	2



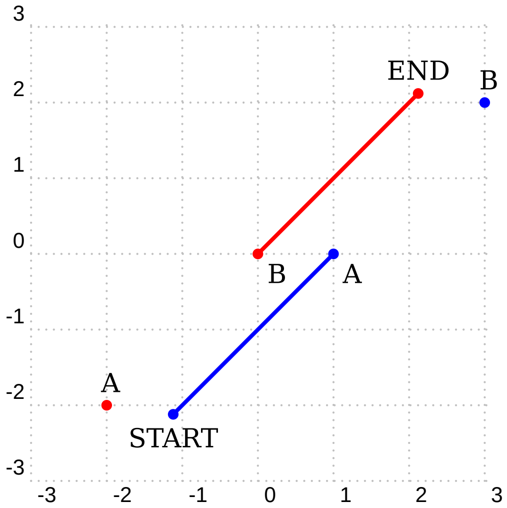

```
\begin{pspicture}[showgrid=true](-3,-3)(3,3)
\psset{linecolor=red}
\Arrows[posStart=2,length=4](-3,-3)(3,3)
\qdisk(-3,-3){2pt}\uput[45](-3,-3){A}
\qdisk(3,3){2pt}\uput[-90](3,3){B}
\psset{linecolor=green}
\Arrows[linewidth=3pt,length=2](0,-3)(0,0.5)
\qdisk(0,-3){2pt}\uput[45](0,-3){A}
\qdisk(0,0.5){2pt}\uput[-90](0,0.5){B}
\psset{linecolor=blue}
\Arrows[linewidth=5pt,linestyle=dashed](3,0)(2,3)
\qdisk(3,0){2pt}\uput[-90](3,0){A}
\qdisk(2,3){2pt}\uput[-90](2,3){B}
\psset{linecolor=black}
\Arrows[posStart=1,linewidth=5pt,linestyle=dotted,
  length=3](-3,2)(1,2)
\qdisk(-3,2){2pt}\uput[-45](-3,2){A}
\qdisk(1,2){2pt}\uput[-90](1,2){B}
\end{pspicture}
```

16 \psOutLine and \psBeforeLine

Syntax:

`\psOutLine [Options] (`*NodeA*`)(`*NodeB*`){`*EndNode*`}`
`\psBeforeLine [Options] (`*NodeA*`)(`*NodeB*`){`*StartNode*`}`

The only special option is `length=`⟨*value*⟩. All other which are possible for `\psline` can be used, too.



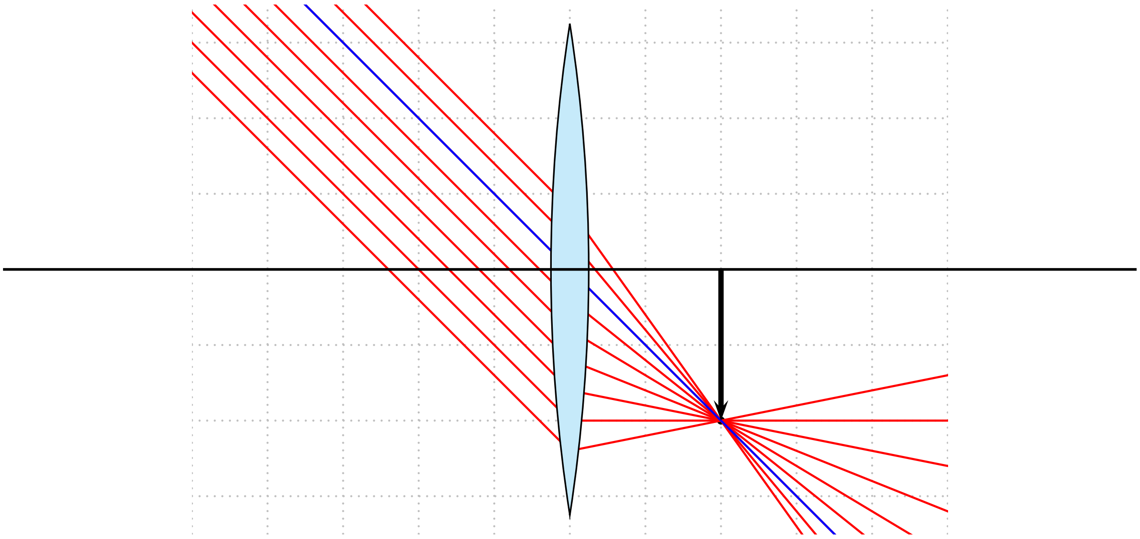

```
\begin{pspicture}[showgrid=true](-3,-3)(3,3)
\psset{linecolor=red, linewidth=1.5pt}
\psOutLine[length=3](-2,-2)(0,0){End}
\qdisk(-2,-2){2pt}\uput[80](-2,-2){A}
\qdisk(0,0){2pt}\uput[-45](0,0){B}
\qdisk(2.12,2.12){2pt}\uput[90](2.12,2.12){END}
%
\psset{linecolor=blue}
\psBeforeLine[length=3](1,0)(3,2){START}
\qdisk(3,2){2pt}\uput[80](3,2){B}
\qdisk(1,0){2pt}\uput[-45](1,0){A}
\qdisk(-1.12,-2.12){2pt}\uput[-90](-1.12,-2.12){START}
\end{pspicture}
```

17 \Parallel

Syntax:

`\Parallel [Options]` (*NodeA*)(*NodeB*)(*StartNode*){*End node*}

The only special option for `\Parallel` is `length=`⟨*value*⟩. The nodes `nodeA` and `nodeB` are known nodes of a given line and `Start node` is the given node of a parallel line. `End node` is the name of the calculated line end.

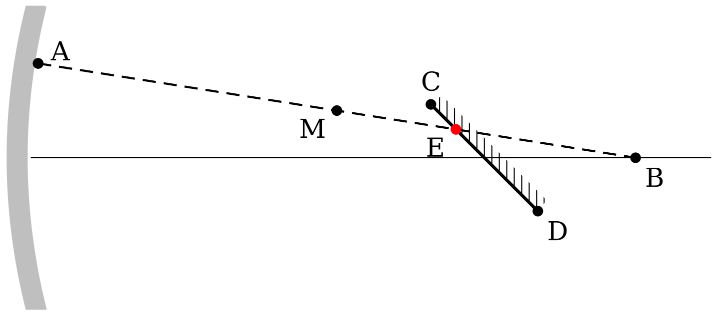


```
\begin{pspicture*}[showgrid=true](-5,-3.5)(5,3.5)
\pnode(2,-2){FF}\qdisk(FF){1.5pt} \pnode(-5,5){A} \pnode(0,0){O}
\multido{\nCountA=-2.4+0.4}{9}{%
  \Parallel[linecolor=red,length=9](O)(A)(0,\nCountA){P1}
  \psline[linecolor=red](0,\nCountA)(FF)
  \psOutLine[linecolor=red,length=9](0,\nCountA)(FF){P2}}
\psline[linecolor=blue](A)(FF)
\psOutLine[linecolor=blue,length=5](A)(FF){END1}
\rput(0,0){\lens[yBottom=-3.5,yTop=3.5,lensGlass=true,lensHeight=6.5,drawing=false,
    spotFi=315,lensWidth=0.5]
  \psline[linewidth=1pt](xLeft)(xRight)
  \psline[length=2,linewidth=2pt,arrows=->](F')(FF)}
\end{pspicture*}
```

18 \ABinterCD and \nodeBetween

This macro is used by the `\telescop` macro. It determines the intersection point of two lines, in this case a ray and the mirror axis. The following figure shows a part of figure 10. Given are the points A, B (focus), C/D (mirror axis). We need the point E to draw the other rays for the ocular, which can be done with the `\ABinterCD` macro. The syntax is:

```
\ABinterCD(A)(B)(C)(D){E}
\nodeBetween(A)(B){C}
```



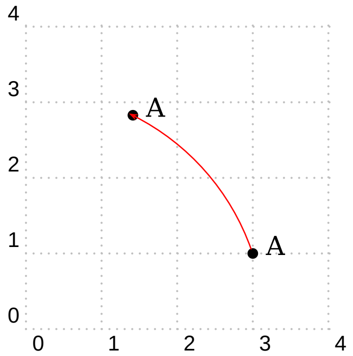

```
\begin{pspicture*}(-0.5,-2.25)(9,2.25)
\rput(0,0){\mirrorCVG[mirrorHeight=4,mirrorWidth=0.25,mirrorDepth=0.25,drawing=false]}
\mirrorCVGRay[mirrorHeight=4,mirrorWidth=0.25,mirrorDepth=0.25,drawing=false](8,1.25)(2,1.25){A}
\psline[linewidth=0.5\pslinewidth](9,0)
\rput{-45}(6,0){\mirrorTwo}
\qdisk(A){2pt}\uput[30](A){A}\pnode(8,0){B}\qdisk(B){2pt}\uput[-45](B){B}
\pnode(! 6 1 45 cos mul sub 1 45 sin mul){C}
\qdisk(C){2pt}\uput[90](C){C}\pnode(! 6 1 45 cos mul add 1 45 sin mul neg){D}
\uput[-45](D){D}\qdisk(D){2pt}\psline[linestyle=dashed](A)(B)
\ABinterCD(A)(B)(C)(D){Inter1}\qdisk(A){2pt}
\nodeBetween(A)(B){M}\qdisk(M){2pt}
{\psset{linecolor=red}
\qdisk(Inter1){2pt}\uput[220](Inter1){E}\uput[220](M){M}}
\end{pspicture*}
```

19 \rotateNode

The syntax is

\rotateNode{*NodeName*}{*Degrees*}

The coordinates of the node A are changed to the new ones. Negative values are possible for rotating clockwise.



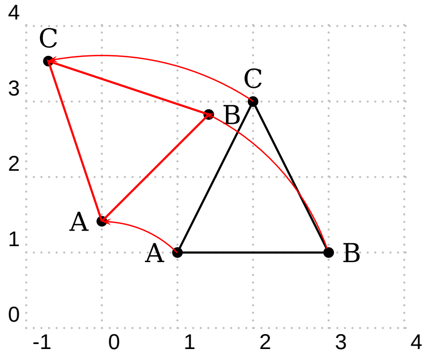

```
\begin{pspicture}[showgrid=true](4,4)
\pnode(3,1){A}\qdisk(A){2pt}\uput[20](A){A}
\rotateNode(A){45}
\qdisk(A){2pt}\uput[20](A){A}
\psarc[linecolor=red,
  linewidth=0.5pt]{->}(0,0){3.16}{19.47}{64.47}
\end{pspicture}
```

20 \rotateTriangle

The syntax is

\rotateNode{*NodeNameA*}{*NodeNameB*}{*NodeNameC*}{*Degrees*}

The coordinates of the nodes `A,B,C` are changed to the new ones. Negative values are possible for rotating clockwise.



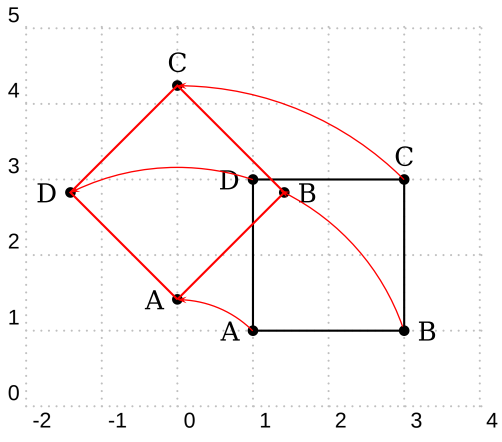

```
\begin{pspicture}[showgrid=true](-1,0)(4,4)
\pnode(1,1){A}\pnode(3,1){B}\pnode(2,3){C}
\qdisk(A){2pt}\uput[180](A){A}\qdisk(B){2pt}\uput[0](B){B}
\qdisk(C){2pt}\uput[90](C){C}
\psline(A)(B)(C)(A) \rotateTriangle(A)(B)(C){45}
\qdisk(A){2pt}\uput[180](A){A}\qdisk(B){2pt}\uput[0](B){B}
\qdisk(C){2pt}\uput[90](C){C}\psline[linecolor=red](A)(B)(C)(A)
\psarc[linecolor=red,linewidth=0.5pt]{->}(0,0){3.16}{19.47}{64.47}
\psarc[linecolor=red,linewidth=0.5pt]{->}(0,0){1.41}{45}{90}
\psarc[linecolor=red,linewidth=0.5pt]{->}(0,0){3.61}{56.31}{101.31}
\end{pspicture}
```

21 \rotateFrame

The syntax is

\rotateFrame{*NodeNameA*}{*NodeNameB*}{*NodeNameC*}{*NodeNameD*}{*Degrees*}

The coordinates of the nodes `A,B,C,D` are changed to the new ones. Negative values are possible for rotating clockwise.



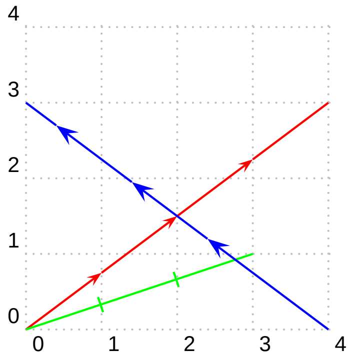

```
\begin{pspicture}[showgrid=true](-2,0)(4,5)
\pnode(1,1){A}\pnode(3,1){B}\pnode(3,3){C}\pnode(1,3){D}
\qdisk(A){2pt}\uput[180](A){A}\qdisk(B){2pt}\uput[0](B){B}
\qdisk(C){2pt}\uput[90](C){C} \qdisk(D){2pt}\uput[180](D){D}
\psline(A)(B)(C)(D)(A)
\rotateFrame(A)(B)(C)(D){45}
\qdisk(A){2pt}\uput[180](A){A}\qdisk(B){2pt}\uput[0](B){B}
\qdisk(C){2pt}\uput[90](C){C} \qdisk(D){2pt}\uput[180](D){D}
\psline[linecolor=red](A)(B)(C)(D)(A)
\psarc[linecolor=red,linewidth=0.5pt]{->}(0,0)
   {3.16}{19.47}{64.47}
\psarc[linecolor=red,linewidth=0.5pt]{->}(0,0){1.41}{45}{90}
\psarc[linecolor=red,linewidth=0.5pt]{->}(0,0){4.24}{45}{90}
\psarc[linecolor=red,linewidth=0.5pt]{->}(0,0)
   {3.16}{71.57}{116.57}
\end{pspicture}
```

22 \arrowLine

The syntax is

\arrowLine `[Options]` (*Start*)(*End*){*ArrowNumber*}

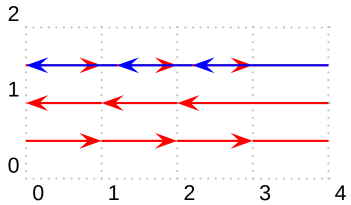
Draws a line from `Start` to `End` with `ArrowNumber` arrows inside.




```
\begin{pspicture}[showgrid=true](4,4)
\arrowLine[linecolor=red](0,0)(4,3){3}
\arrowLine[linecolor=green,arrowsize=6pt,arrows=-|](0,0)(3,1){2}
\arrowLine[linecolor=blue,arrowOffset=0.75,arrowsize=6pt](4,0)(0,3){3}
\end{pspicture}
```

22.1 Options

A special option is `arrowOffset`, which makes it possible to draw lines with different arrows. By default the arrows are placed symetrically. This can be moved by `arrowOffset`. Additionally all other valid options for pslines are possible her, too.




```
\begin{pspicture}[showgrid=true](4,2)
\arrowLine[arrowsize=6pt,linecolor=red](0,0.5)(4,0.5){3}
\arrowLine[arrowsize=6pt,linecolor=red,
  arrows=<-](0,1)(4,1){3}
\arrowLine[arrowsize=6pt,linecolor=red](0,1.5)(4,1.5){3}
\arrowLine[arrowsize=6pt,linecolor=blue,arrows=<-,
  arrowOffset=0.2](0,1.5)(4,1.5){3}
\end{pspicture}
```

23 List of all optional arguments for pst-optic

Key	Type	Default
lensTwo	boolean	false
lensGlass	boolean	true
drawing	boolean	true
rightRay	boolean	false
xLeft	ordinary	-7.5
xRight	ordinary	7.5
yBottom	ordinary	-3.0
yTop	ordinary	3.0
lensType	ordinary	CVG
lensColor	ordinary	lightgray
lensWidth	ordinary	0.5
lensDepth	ordinary	1
lensHeight	ordinary	5
lensScale	ordinary	1
lensArrowSize	ordinary	0.2
lensArrowInset	ordinary	0.5
mirrorType	ordinary	CVG
mirrorDepth	ordinary	1
mirrorHeight	ordinary	5
mirrorWidth	ordinary	0.25
mirrorColor	ordinary	lightgray
mirrorFocus	ordinary	8
posMirrorTwo	ordinary	6
mirrorTwoAngle	ordinary	45
refractA	ordinary	1
refractB	ordinary	1.41
XO	ordinary	0
YO	ordinary	0
posStart	ordinary	0
length	ordinary	2
focus	ordinary	2
AB	ordinary	1
OA	ordinary	-3
arrowOffset	ordinary	0
nameA	ordinary	A
spotA	ordinary	270
nameB	ordinary	B
spotB	ordinary	90
nameF	ordinary	F
spotF	ordinary	270
nameO	ordinary	O
spotO	ordinary	225
nameAi	ordinary	A’

Continued on next page

Continued from previous page

Key	Type	Default
spotAi	ordinary	90
nameBi	ordinary	B'
spotBi	ordinary	270
nameFi	ordinary	F'
spotFi	ordinary	270
rayColor	ordinary	red
rayWidth	ordinary	1.5\pslinewidth
AnglePrism	ordinary	[none]
AnglePlan1	ordinary	[none]
AnglePlan2	ordinary	[none]
lambda	ordinary	[none]
k	ordinary	[none]
notations	boolean	true

References

[1] Denis Girou and Manuel Luque. *PST-lens - PostScript macros for Generic TeX*. ftp://ftp.dante.de/tex-archive/graphics/pstricks/contrib/pst-lens/, 2001.

[2] Michel Goosens, Frank Mittelbach, Serbastian Rahtz, Denis Roegel, and Herbert Voß. *The LaTeX Graphics Companion*. Addison-Wesley Publishing Company, Reading, Mass., 2nd edition, 2007.

[3] Laura E. Jackson and Herbert Voß. Die Plot-Funktionen von pst-plot. *Die TeXnische Komödie*, 2/02:27–34, June 2002.

[4] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*. IWT, Vaterstetten, 1989.

[5] Manuel Luque. *Vue en 3D*. http://members.aol.com/Mluque5130/vue3d16112002.zip, 2002.

[6] Herbert Voß. Die mathematischen Funktionen von Postscript. *Die TeXnische Komödie*, 1/02:40–47, March 2002.

[7] Herbert Voss. *PSTricks Support for pdf*. http://PSTricks.de/pdf/pdfoutput.phtml, 2002.

[8] Herbert Voß. *LaTeX in Mathematik und Naturwissenschaften*. Franzis-Verlag, Poing, 2006.

[9] Herbert Voß. *PSTricks – Grafik für TeX und LaTeX*. DANTE – Lehmanns, Heidelberg/Hamburg, 5. edition, 2008.

[10] Michael Wiedmann and Peter Karp. *References for TeX and Friends*. http://www.miwie.org/tex-refs/, 2003.

[11] Timothy Van Zandt. *PSTricks - PostScript macros for Generic TeX*. http://www.tug.org/application/PSTricks, 1993.

Index