

Lenses and Mirrors: PST-optic v0.95

Manuel Luque* Herbert Voß†

July 21, 2006

Abstract

This version uses the extended keyval package **xkeyval**, so be sure that you have installed this package together with the special one **pst-xkey** for PSTricks. The **xkeyval** package bundle is available at <ftp://ftp.dante.de/pub/tex/macros/latex/contrib/xkeyval/>. It is also important that after **pst-optic** no package is loaded, which uses the old keyval interface.

*Mluque5130@aol.com

†voss@perce.de

Contents

I	General Options	5
1	<code>\resetOpticOptions</code>	6
II	Lenses	7
2	The Coordinates of the predefined Nodes	7
3	The Lens Type	7
4	<code>\Transform</code>	9
5	<code>\rayInterLens</code>	11
6	<code>\telescope</code>	12
III	Mirrors	13
7	options	13
8	<code>\mirrorCVG</code>	13
9	<code>\mirrorDVG</code>	14
9.1	Drawing Rays in the Mirror Macros	15
9.2	<code>\planMirrorRay</code>	15
9.3	<code>\symPlan</code>	16
10	Beam Light	16
IV	Refraction	19
11	<code>\refractionRay</code>	19
12	Total Reflection	20

V	Spherical Optic	21
13	\lensSPH	21
13.1	Convergent Lens	21
13.2	Divergent Lens	21
13.3	Options	22
14	\mirrorCVG	22
15	\mirrorDVG	23
16	\ABinterSPHLens	23
17	\lensSPHray	24
18	\reflectionRay	25
19	Refraction at a Spherical Surface	27
19.1	Simple Example	27
19.2	Height of an Image	28
VI	Utility Macros	29
20	\eye	29
21	\Arrows	29
22	\psOutLine	30
23	\psBeforeLine	31
24	\Parallel	31
25	\ABinterCD	32
26	\nodeBetween	33
27	\rotateNode	33
28	\rotateTriangle	33
pst-optic-doc.tex		3

29 \rotateFrame	34
30 \arrowLine	35
30.1 Options	35
31 Credits	36

Introduction

`pstricks` writes pure PostScript^[2] code, so it is not possible to run T_EX files with pdfL_AT_EX when there are `pstricks` macros in the document. If you still need a PDF output use the package `pdftricks.sty`^[4] or the for Linux free available program `vlatex` (<http://www.micropress-inc.com/linux/>) or build the PDF with `ps2pdf` (dvi→ps→pdf).

If you need package `graphicx.sty` load it before any `pstricks` package. You do not need to load `pstricks.sty`, it will be done by `pst-optic.sty` by default.

Part I

General Options

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

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

Table 1: General options and the defaults

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

```

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

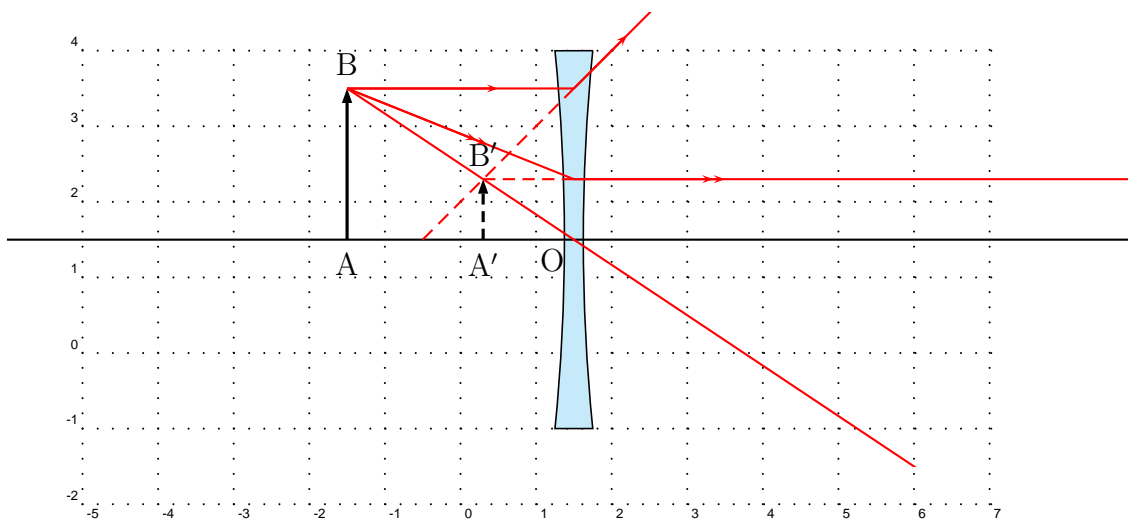
```

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

```

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

```



1 `\resetOpticOptions`

This macro resets all `pst-optic` options to the default value.

Part II

Lenses

There are macros for the convergent and divergent lens

`\lens[CVG]` Convergent (Collecting lens) - default

`\lens[DVG]` Divergent (Scatter lens)

2 The Coordinates of the predefined Nodes

Figure 1 shows the coordinates of the predefined nodes (see table 1).

```

1 \begin{pspicture}*(-8,-3.25)(8,3.25)
2   \rput(0,0){%
3     \lens[drawing=false]
4     \psline[linewidth=1pt](xLeft)(xRight)
5     \qdisk(A){1.5pt}
6     \qdisk(B){1.5pt}
7     \qdisk(A'){1.5pt}\qdisk(B'){1.5pt}
8     \qdisk(F){1.5pt}\qdisk(F'){1.5pt}
9     \qdisk(O){1.5pt}\qdisk(I){1.5pt}
10    \qdisk(I'){1.5pt}\qdisk(I1){1.5pt}
11    \qdisk(I2){1.5pt}
12    \uput[270](A){A}\uput[90](B){B}
13    \uput[270](F){F}\uput[0](I){I}
14    \uput[0](I'){\mathrm{I'}}\uput[270](F'){\mathrm{F'}}
15    \uput[270](B'){\mathrm{B'}}\uput[90](A'){\mathrm{A'}}
16    \uput[180](I1){I1}\uput[0](I2){I2}%
17  }
18 \end{pspicture}

```

3 The Lens Type

Using `\lens[<lensType>]` gives the in figure 2 and 3 shown lenses with the default values from table 2.

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

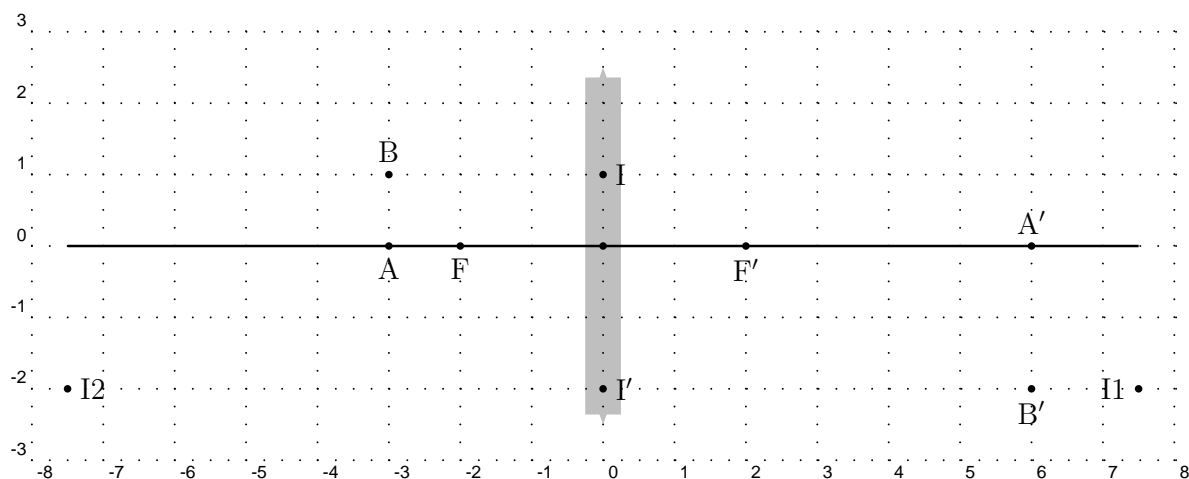


Figure 1: Coordinates of the predefined Nodes

Option	Name	Default
Lense type	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{OA}	OA	-4
Distance \overline{AB}	AB	1.5
Lens color	lenscolor	black
Arrow length in cm	lensarrowsize	0.2
Arrow inset in cm	lensarrowinset	0.5

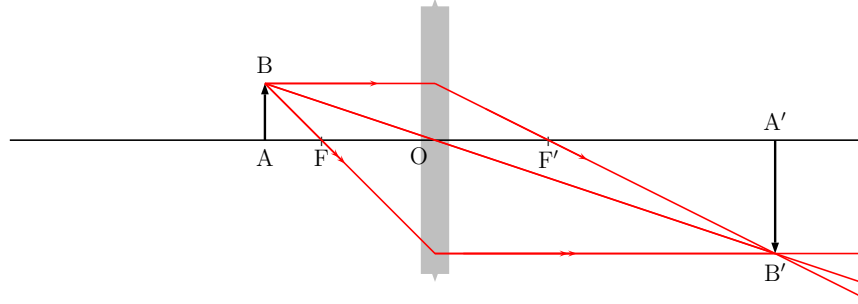
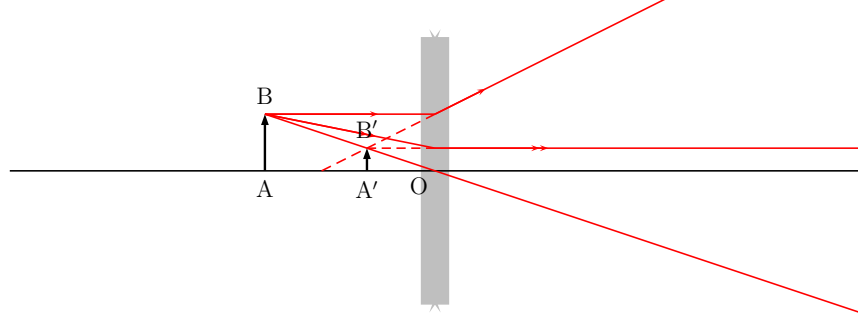
Table 2: Available options for lenses with the defaults

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

```

1 \begin{pspicture}*(-7.5,-3)(7.5,3)
2   \rput(0,0){\lens[...]}
3 \end{pspicture}
```

The star version enables the clipping option.

Figure 2: `\lens[lensType=CVG]` (Collecting lens)Figure 3: `\lens[lensType=DVG,focus=-2,spotAi=270,spotBi=90]` (Scatter lens)

4 `\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 4).

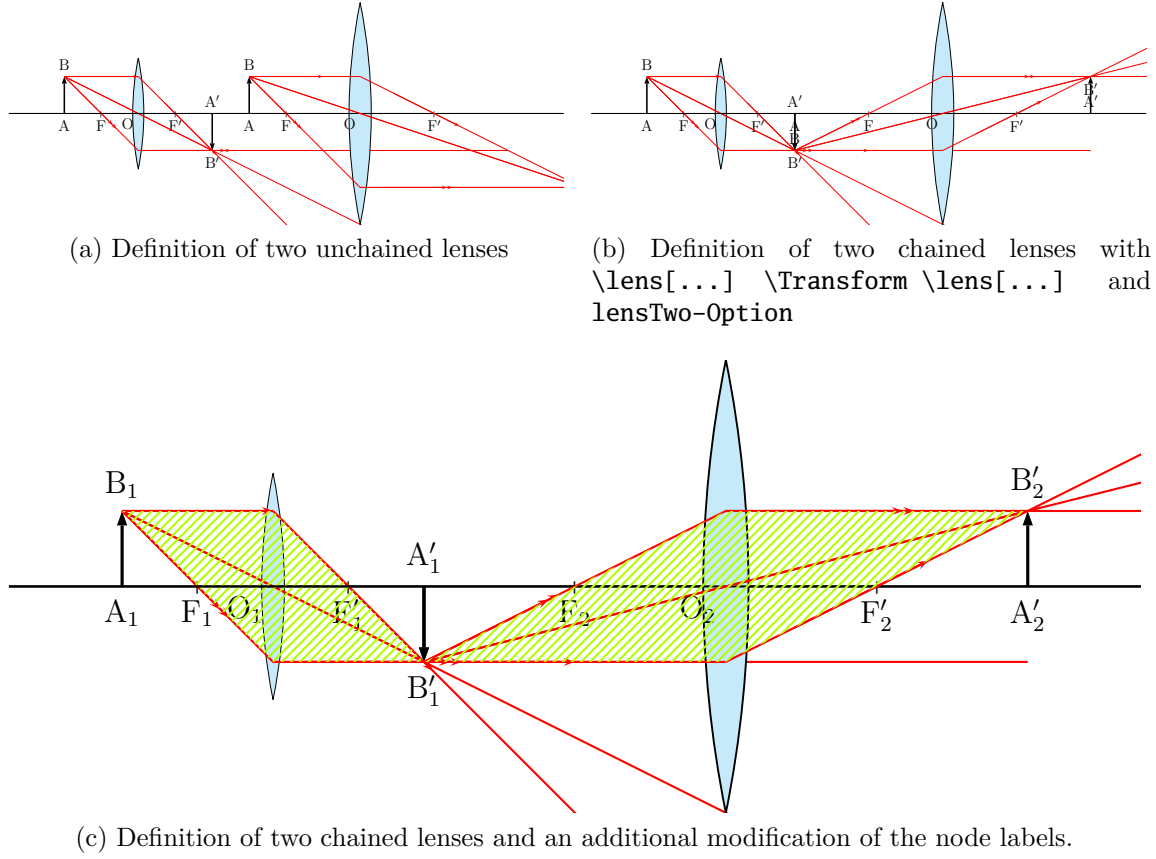
Alt	A	B	A'	B'	O	F	F'	I	I'	XO	YO	OA'	A'B'
Neu	A1	B1	A'1	B'1	O1	F1	F'1	I1	I'1	XO1	YO1	O1A1'	A'1B'1

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

```

1 \begin{pspicture}*(-7.5,-3)(7.5,3)
2 \rput(0,0){%
3   \lens[lensScale=0.6,X0=-4,%
4     nameF=F_1,nameA=A_1,nameB=B_1,%
5     nameFi=F'_1,nameAi={ },nameBi={ },nameO=O_1,

```

Figure 4: The meaning of the `\Transform`-Macro with the default labels

```

6   focus=1,OA=-2,lensGlass=true, lensWidth=0.5]%
7   }
8   \pspolygon[style=rayuresJaunes,linestyle=none](B)(I)(B')(I')(B)
9   \Transform
10  \rput(0,0){%
11    \lens[lensScale=1.2,X0=2,focus=2,%
12      nameA=A'_1,spotA=90,nameB=B'_1,spotB=270,%
13      nameO=O_2,nameAi=A'_2,spotAi=270,%
14      nameBi=B'_2,spotBi=90,nameF=F_2,nameFi=F'_2,%
15      lensTwo=true,%
16      lensGlass=true,lensWidth=0.5]%
17  }
18  \pspolygon[style=rayuresJaunes,linestyle=none](B)(I)(B')(I')(B)
19  \end{pspicture}

```

5 \rayInterLens

This macro is only useful for a two-lens-system. Figure 5 shows such a system. The nodes **B1**, **I11**, **F'1**, **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. **\rayInterLens** defines such nodes. The Syntax:

\rayInterLens(StartNode)(IntermediatNode)(LensDistance){LensNode}

For the node of figure 5 we have

```
1 \rayInterLens(I11)(B'1){4}{Inter1L2}
2 \psline(B1)(I11)(B'1)(Inter1L2)
3 \rayInterLens(O1)(B'1){4}{Inter2L2}
4 \psline(B1)(O1)(B'1)(Inter2L2)
```

The two parallel lines are drawn with the **\Parallel**-Macro.

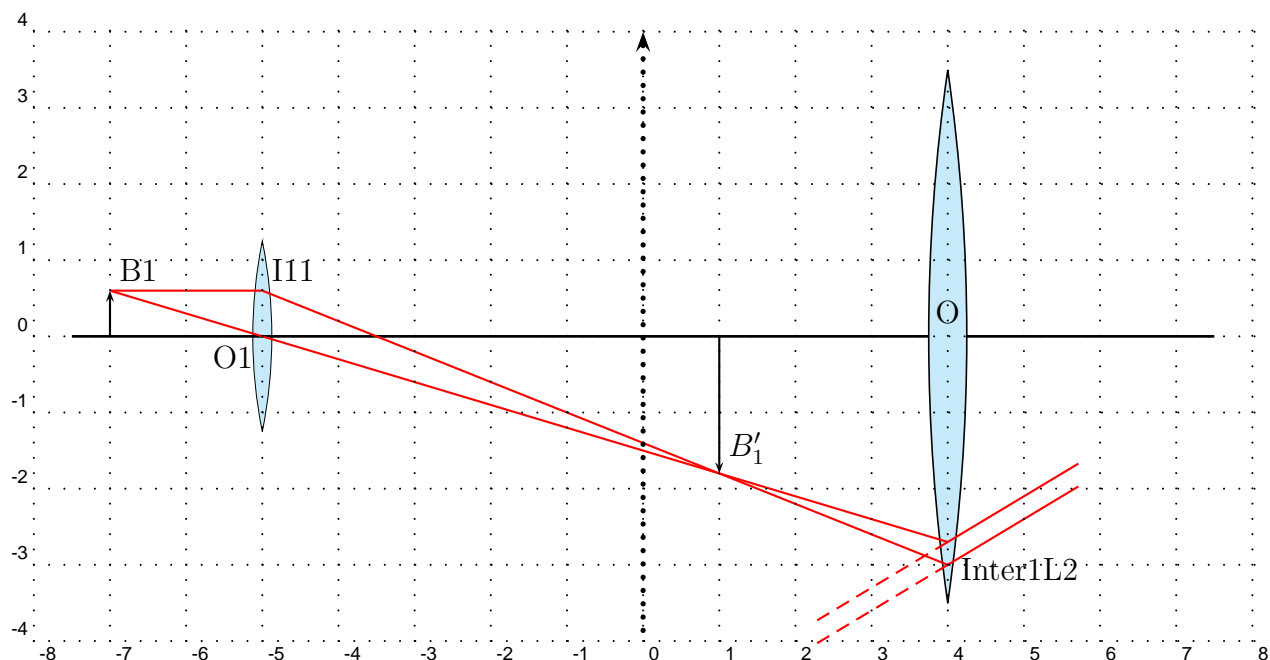


Figure 5: Demonstration of **\rayInterLens**

6 \backslash telescope

Figure 6 shows the configuration of a telescope and table 4 the special options for the \backslash telescop-Macro.

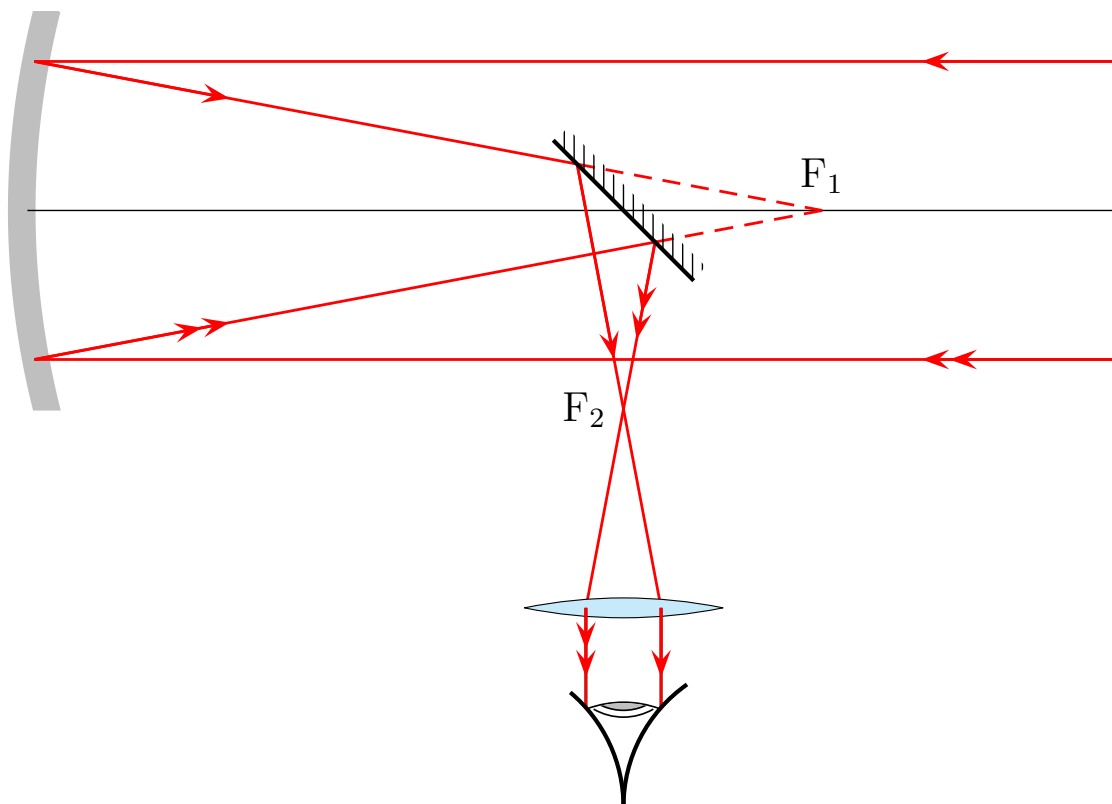


Figure 6: \backslash telescope-Macro

Part III

Mirrors

7 options

Figure 7 shows the available mirrors and table 4 the possible options.

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

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

8 `\mirrorCVG`

Figure 8 shows the default for the `mirrorCVG`-macro with the predefined nodes and three default rays.

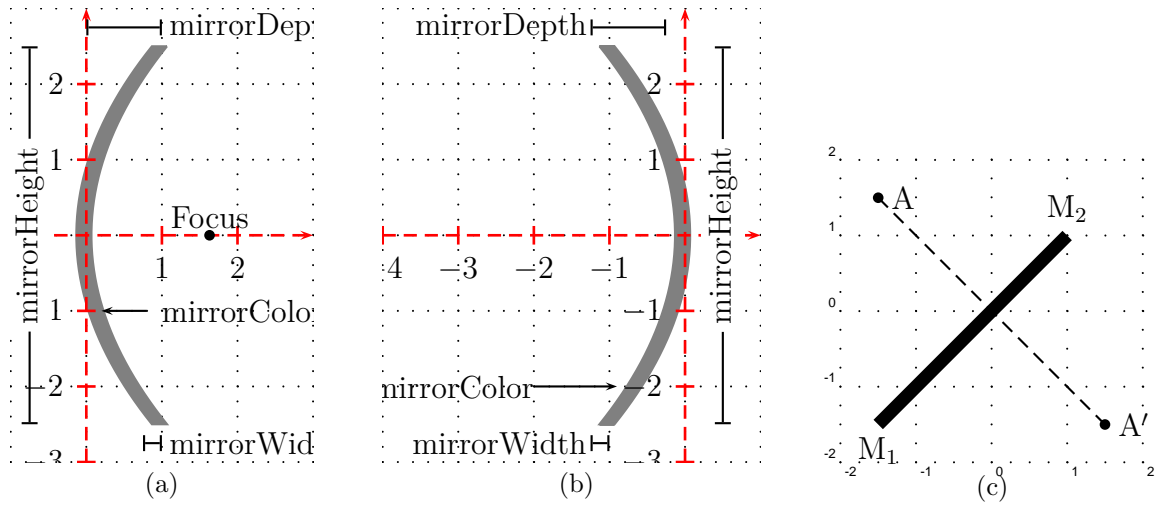


Figure 7: The different mirror macros: a) `\mirrorCVG` b) `\mirrorDVG` c) `\planMirrorRay`

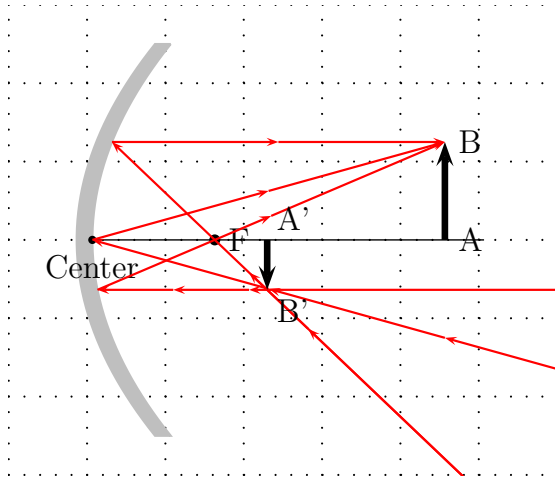


Figure 8: Parabolic Mirror `\mirrorCVG`

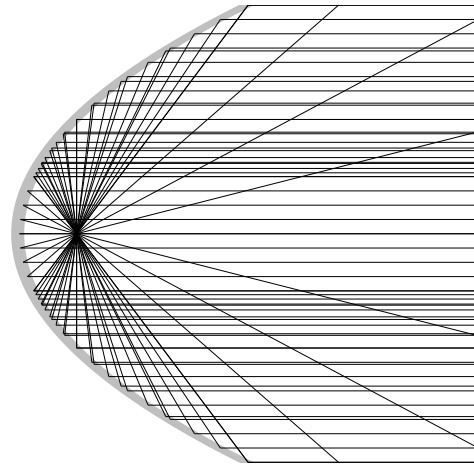


Figure 9: Example

9 `\mirrorDVG`

Figure 10 shows the defaults for the macro `\mirrorDVG`-Makros.

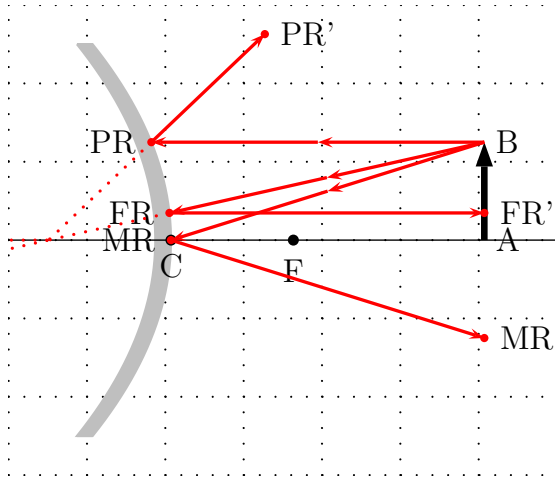


Figure 10: \mirrorDVG

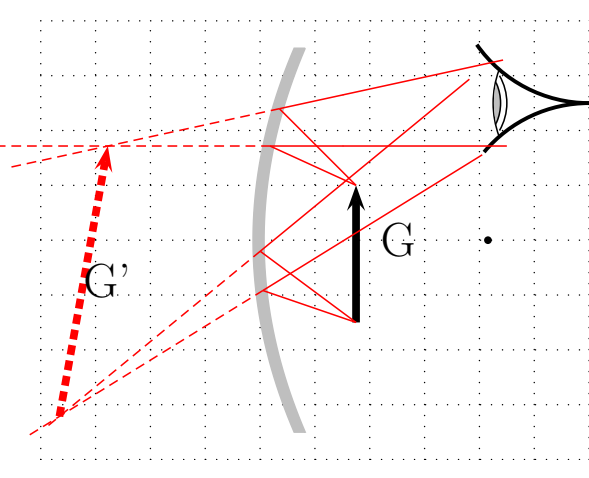


Figure 11: Example as a magnifier

9.1 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.

9.2 \planMirrorRay

The **planMirrorRay**-Macro calculates the coordinates of a mirrored point. In figure 7c 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.

9.3 `\symPlan`

`\symPlan` allows to mirroring complete plain graphical objects along a virtual center line. Figure 12 shows that this mirroring is a mathematical one and not a physical one. For more examples look at [3]. The syntax is:

`\symPlan(node1)(node2){Graphicobject}`

The two nodes define the mirror axis and the graphics object is in most cases a user defined macro, f.ex:

```

1 \newcommand{\dtk}{%
2   \pstextpath(0,0){%
3     \psplot[linestyle=none]{0}{8}{x sqrt sqrt 2 mul}}%
4     {\Large Die \TeX{}nische Komödie von DANTE}%
5   }
6 \begin{pspicture}(-4.5,-2)(2.5,5)
7   \pnode(-4,-2){M1} \uput[-90](M1){M1}
8   \pnode(4,4){M2} \uput[90](M2){M2}
9   \psline[linewidth=5\pslinewidth,linestyle=lightgray](M1)(M2)
10  \rput(-3.5,-1.75){\dtk}% Original schreiben
11  \symPlan(M1)(M2){\rput(-3.5,-1.75){\dtk}}% Spiegelbild schreiben
12 \end{pspicture}
```

This example needs the package `pst-text.sty` for the `\pstextpath` macro (CTAN:/graphics/pstricks/generic/pst-text.tex).

10 Beam Light

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

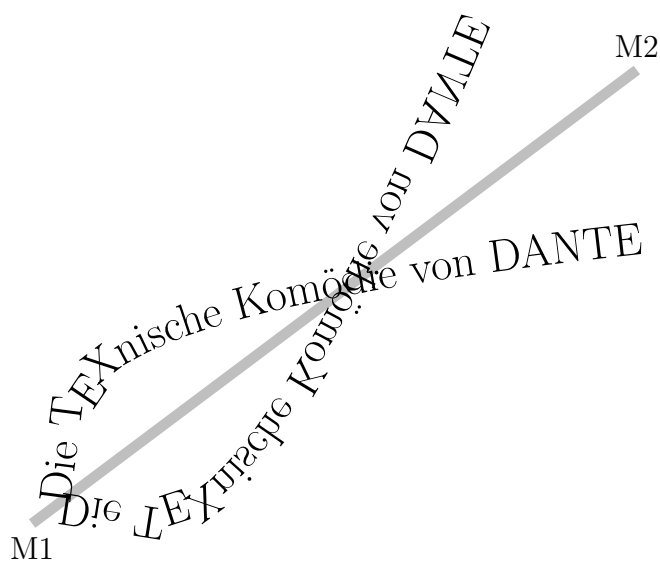
`\beamLight[<Options>]`

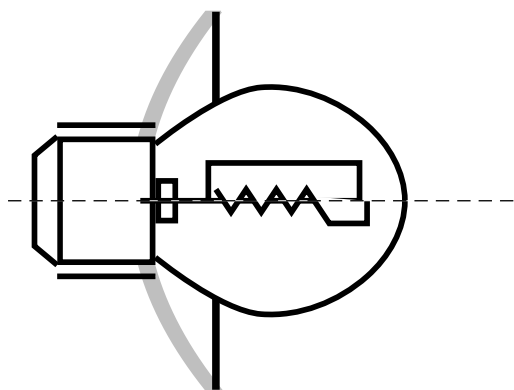
The predefined options especially for the `pspicture`-coordinates are

```

1 \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.

Figure 12: Demonstration of the `\symPlan`-Macro

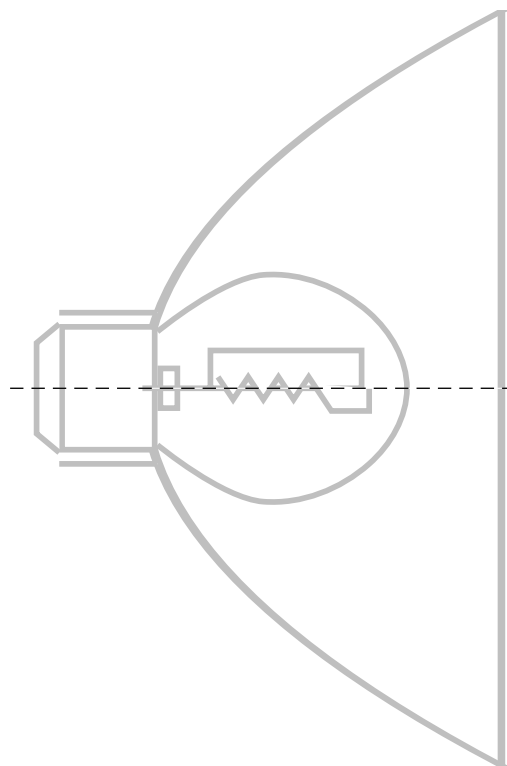


```

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

```

Figure 13: `\beamLight` without any Options



```

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

```

Figure 14: `\beamLight` with Options

Part IV

Refraction

11 \refractionRay

The syntax is

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

The macro uses the law of Snell

$$\frac{n_1}{n_2} = \frac{\sin \beta}{\sin \alpha} \quad (1)$$

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

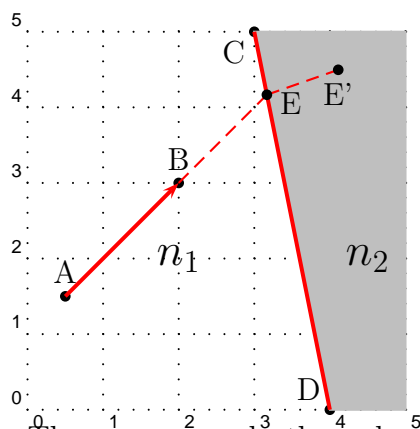
$$n_1 = 1 \quad (2)$$

$$n_2 = 1.41 \quad (3)$$

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

The refraction numbers 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 refraction number. This happens when $\sin \beta > 1$ in equ. 1. In this case we have $\alpha = \beta$, a total reflection.



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

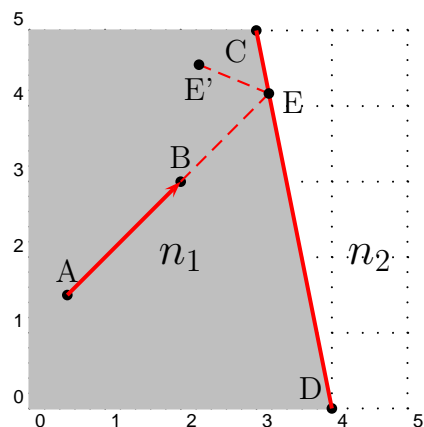
```
\refractionRay(A)(B)(C)(D){1}{4}{E}
```

The macro needs the values for the four nodes, the two refraction numbers and the name for the end node. As you

$$n_1 < n_2 \quad (4)$$

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.

12 Total Reflection



In the figure the macro was called as

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

$$n_1 > n_2 \quad (5)$$

Part V

Spherical Optic

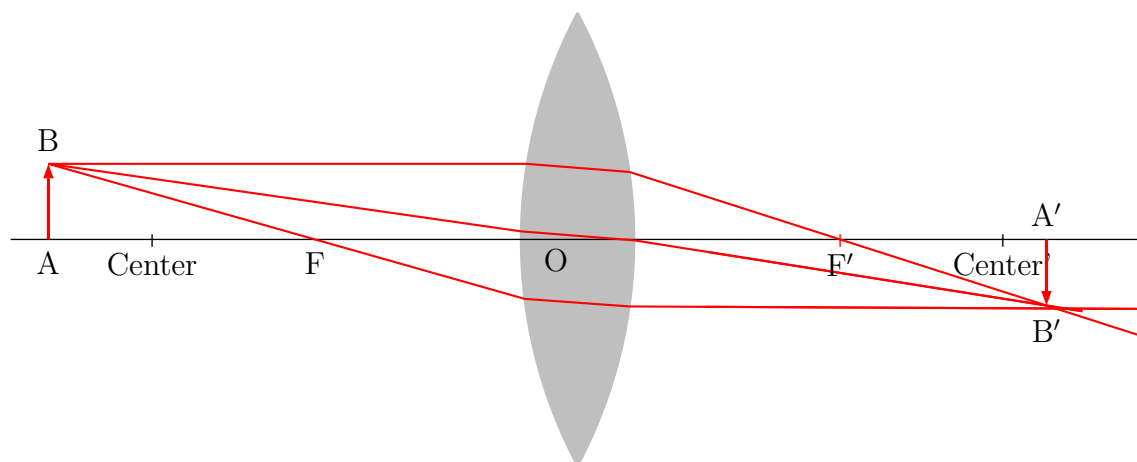
13 `\lensSPH`

13.1 Convergent Lens

The syntax is

```
\lensSPH[<Options>]
\lensSPH[lensType=CVG,<Options>]
```

Without any option it draws a spherical convergent lens:



It changes some default values for the options to:

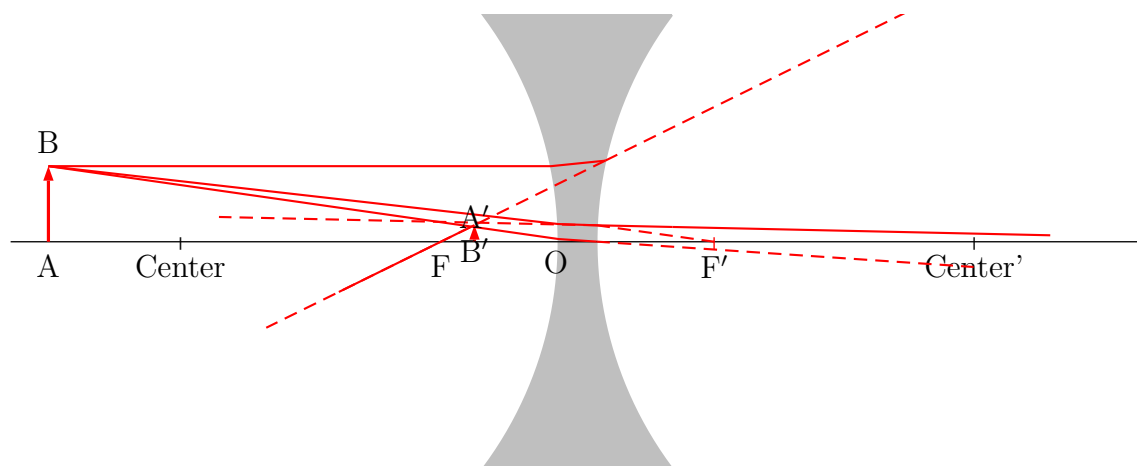
Meaning	Name	Default
Object Distance in cm	OA	-7
Lens Height in cm	lensHeight	6
Lens Width in cm	lensWidth	1.5
Refraction Number n_2	refractB	2

13.2 Divergent Lens

The syntax is

```
\lensSPH[lensType=DVG,<Options>]
```

It draws a spherical divergent lens:



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

13.3 Options

The macro uses the law of Snell

$$\frac{n_1}{n_2} = \frac{\sin \beta}{\sin \alpha} \quad (6)$$

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

$$n_1 = 1 \quad (7)$$

$$n_2 = 1.41 \quad (8)$$

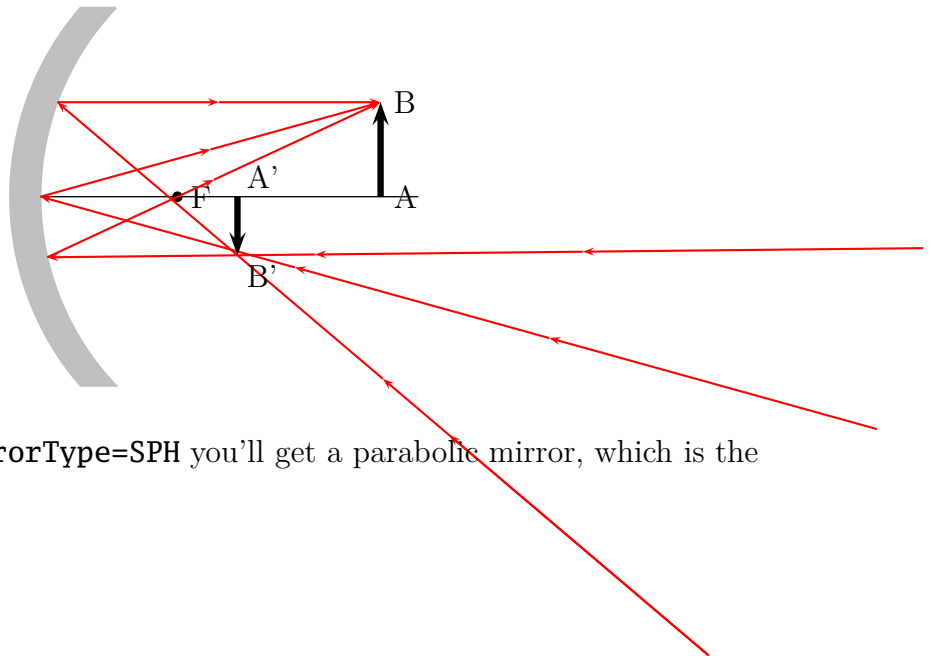
and α the incoming and β the outgoing angle of the ray.

The refraction numbers have the internal names `refractA` and `refractB`.

14 `\mirrorCVG`

The syntax is

`\mirrorCVG[mirrorType=SPH]`

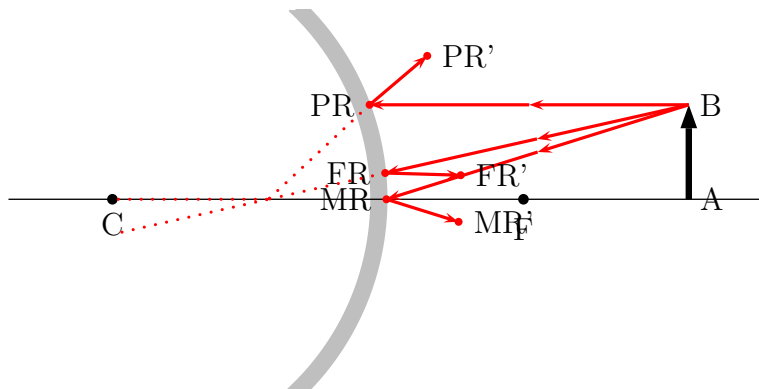


Without the option `mirrorType=SPH` you'll get a parabolic mirror, which is the default.

15 `\mirrorDVG`

The syntax is

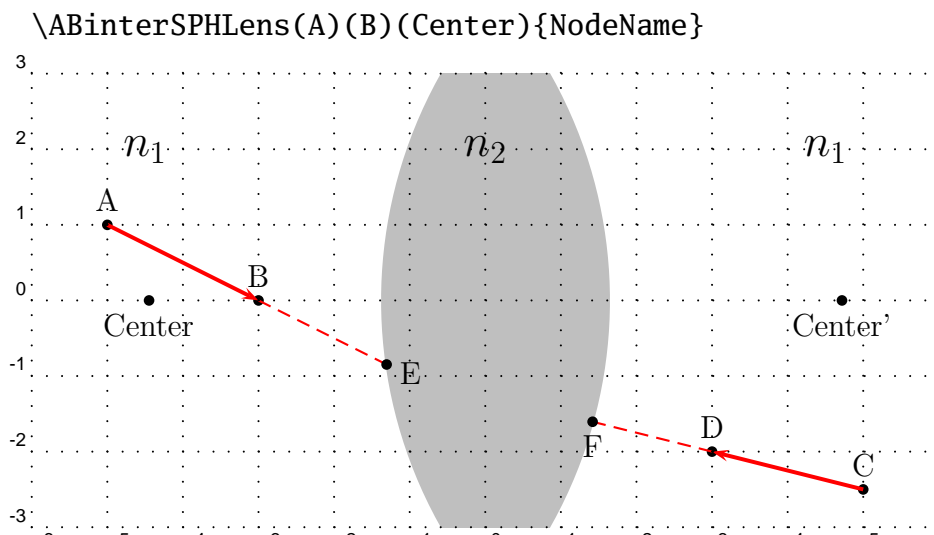
```
\mirrorDVG[mirrorType=SPH]
```



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

16 `\ABinterSPHLens`

The syntax is



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}`

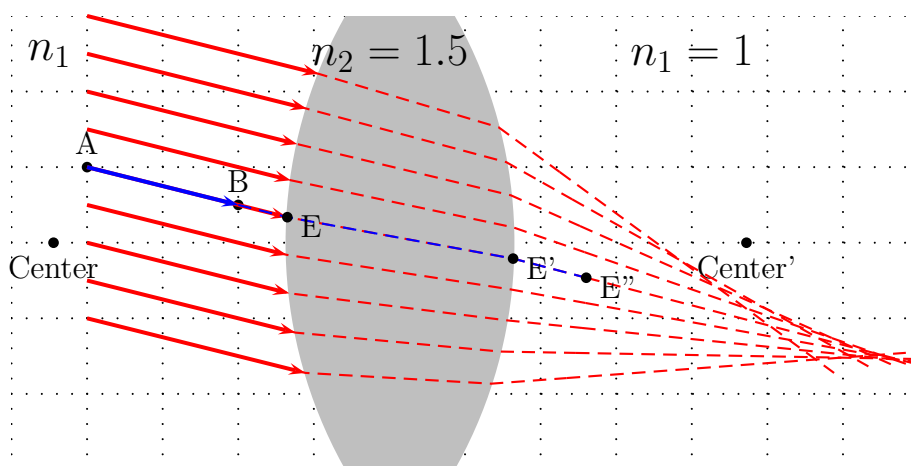
17 `\lensSPHray`

The syntax is

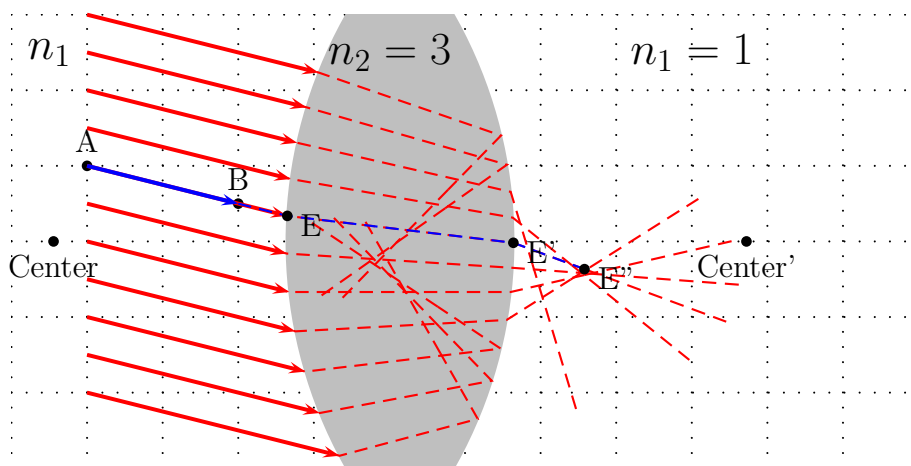
`\lensSPHray[Option](A)(B){refractA}{refractB}{NodeName}`

This macro calculates the coordinates of the given ray \overline{AB} on its way into the lens. The only possible option `ightRay=false|true`² enables rays from the right to the left. There are still some problems with this option but try it out.

²Default is `false`



And the same with $n_2 = 3$:

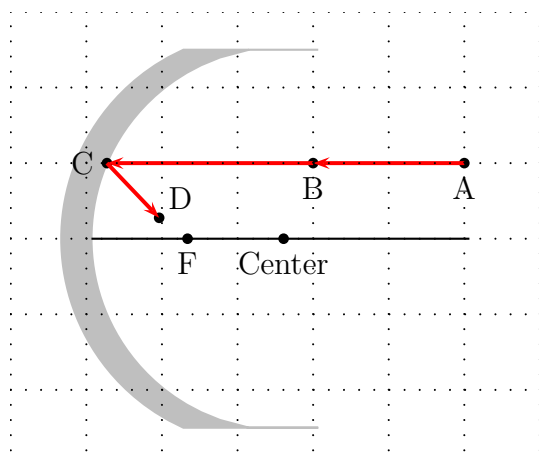


18 \reflectionRay

The syntax is

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

This macro calculates the coordinates of the given ray \overline{AB} on its way out of the mirror. The only senseful option is `mirrorType=CVG|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.



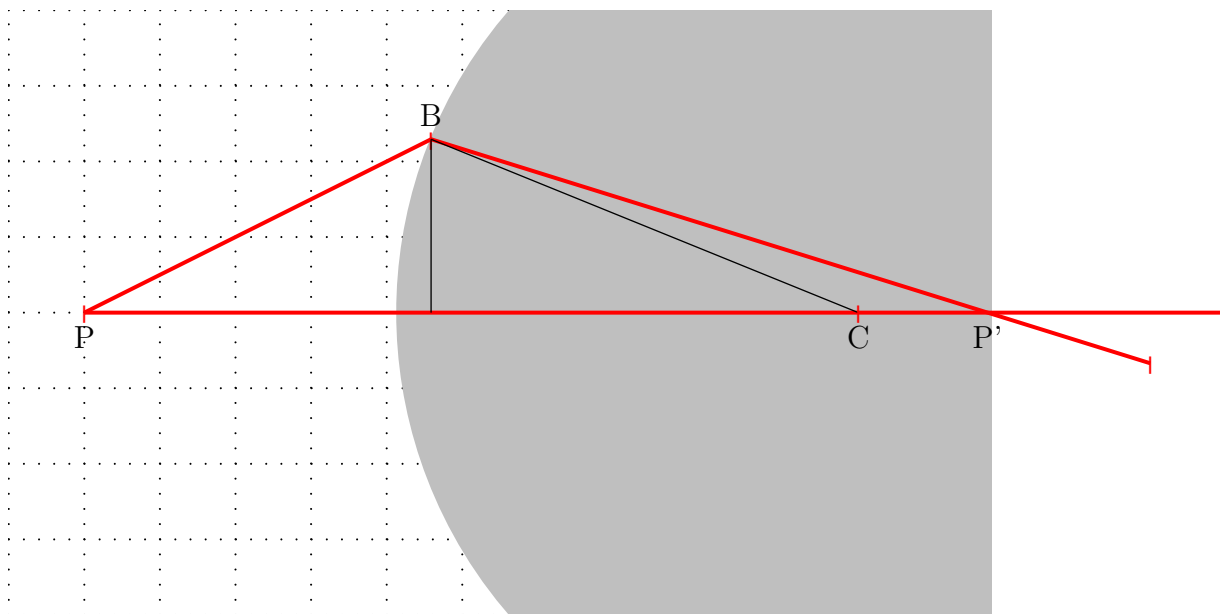
```

1 \begin{pspicture*}(-1,-3)(6,3)
2   \rput(0,0){%
3     \mirrorCVG[%
4       mirrorType=SPH,%
5       mirrorHeight=5, mirrorWidth=0.2,%
6       yBottom=-3,yTop=3,drawing=false,%
7       mirrorDepth=3]%
8     \qdisk(Center){2pt}\qdisk(Focus){2pt}
9     \uput[-90](Center){Center}\uput[-90](Focus){F}
10    \psline(0)(xRight)
11  }
12  \ABinterSPHLens(5,1)(3,1)(Center){C}
13  \reflectionRay[mirrorType=CVG-SPH](5,1)(C){D}
14  \qdisk(5,1){2pt}\uput[-90](5,1){A}
15  \qdisk(3,1){2pt}\uput[-90](3,1){B}
16  \qdisk(C){2pt}\uput[180](C){C}
17  \qdisk(D){2pt}\uput[45](D){D}
18  \psset{linewidth=1.5pt,linecolor=red,arrows=->}
19  \psline(5,1)(3,1)
20  \psline(3,1)(C)
21  \psline(C)(D)
22  \psgrid
23 \end{pspicture*}

```

19 Refraction at a Spherical Surface

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



```

1  \begin{pspicture}*(-10,-4)(3,4)
2  \psgrid[subgriddiv=0,griddots=5,gridlabels=7pt]
3  \rput(0,0){\lensSPH[%
4    lensType=CVG,%
5    lensHeight=12,lensWidth=10,%
6    yBottom=-4,yTop=4,xLeft=-5,xRight=5,drawing=false]}
7  \psset{linecolor=red,linewidth=1.5pt,dotstyle=|}
8  \pnode(-9,0){P}\psdots(P)\uput[-90](P){P}
9  \psline(P)(xRight)
10 \lensSPHray(P)(-5,2){1}{9}{Q}%
11 \psline(P)(Q)(Q')
12 \psdots(Q)\uput[90](Q){B}
13 \ABinterCD(Q)(Q')(0,0)(5,0){P'}
14 \psdots(Q')\uput[-90](P'){P'}
15 \psline[linewidth=0.5pt,linecolor=black](Center')(Q)
16 \psline[linewidth=0.5pt,linecolor=black](Q)(Q|0,0)
17 \psdots(Center')\uput[-90](Center'){C}
18 \end{pspicture}

```


Part VI

Utility Macros

20 `\eye`

Syntax:

`\eye`

There are no Options for this symbol of an human eye (figure 15).

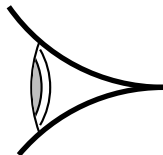


Figure 15: The `\eye`-Macro

Use the `\rput`-macro to put the eye elsewhere:

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

21 `\Arrows`

Syntax:

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

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

Table 5: Options for the `\Arrows`-Macro

The code for figure 16:

```
1 \Arrows[posStart=2,length=4](-3,-3)(3,3)
2 \Arrows[linewidth=3pt,length=2](0,-3)(0,0.5)
3 \Arrows[linewidth=5pt,linestyle=dashed](3,0)(2,3)
4 \Arrows[posStart=1,linewidth=5pt,linestyle=dotted,length=4](-3,2)(1,2)
```

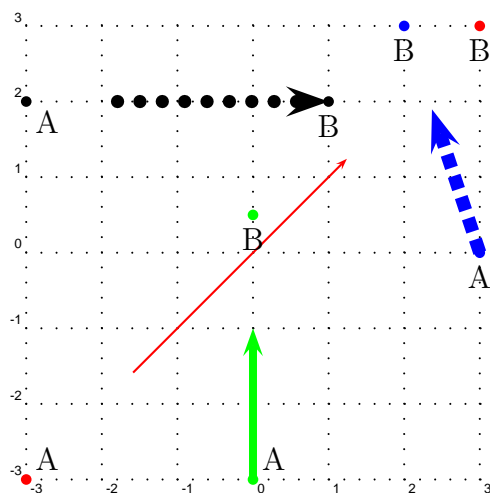


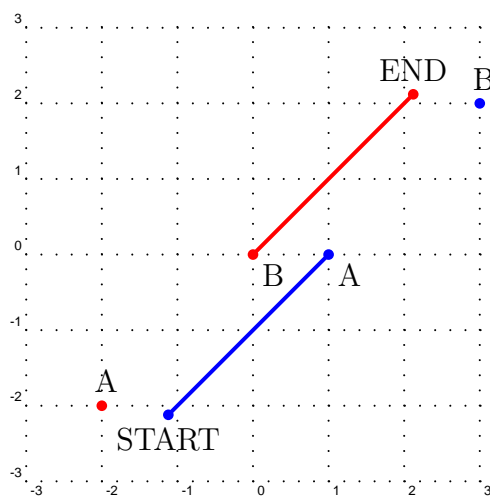
Figure 16: Arrows Demo

22 `\psOutline`

Syntax:

`\psOutline[Options](NodeA)(NodeB){EndNode}`

The only special option is `length=<avalue>`. All other which are possible for `\psline` can be used, too.

Figure 17: `psOutline` and `psBeforeLine` Demo

The code for figure 17:

```
1 \psOutLine[length=3](-2,-2)(0,0){End}
```

23 `\psBeforeLine`

Syntax:

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

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

The code for figure 17:

```
1 \psBeforeLine[length=3](0,0)(2,2){START}
```

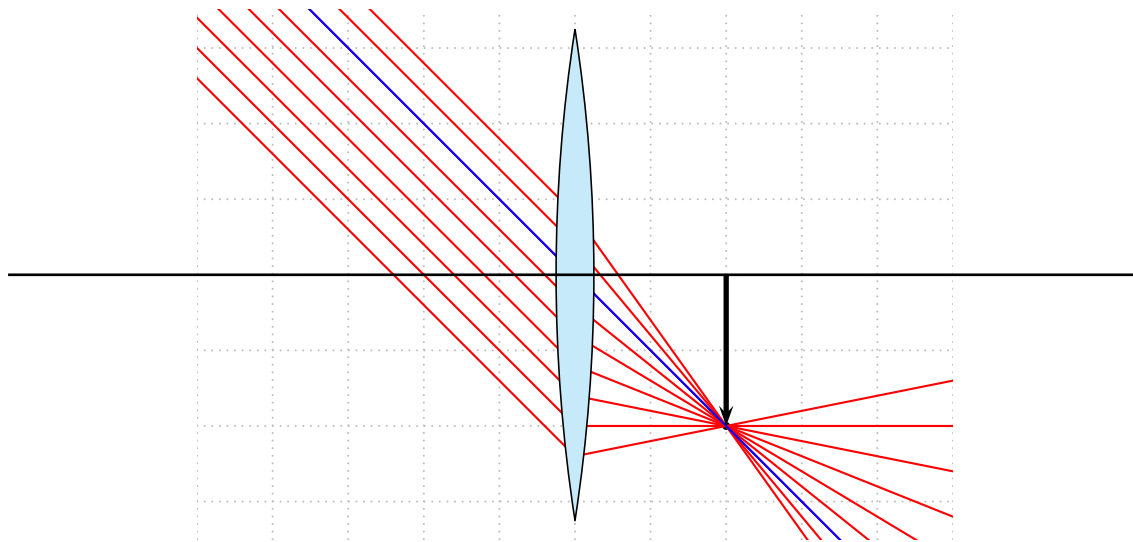
24 `\Parallel`

Syntax:

```
\Parallel[Options](NodeA)(NodeB)(Start node){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. The use of `Parallel` is shown for an example (figure 18).

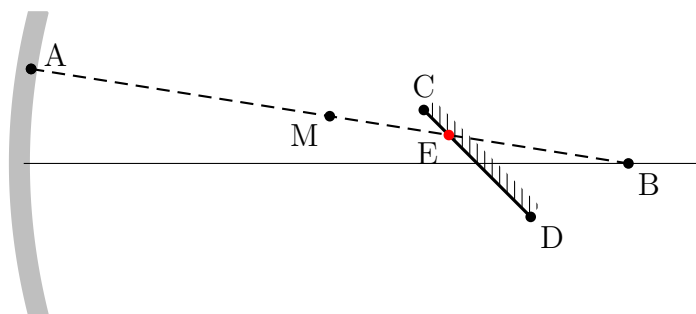
```
1 \begin{pspicture*}[showgrid=true](-5,-3.5)(5,3.5)
2 \pnode(2,-2){FF}\qdisk(FF){1.5pt}
3 \pnode(-5,5){A}
4 \pnode(0,0){O}
5 \multido{\nCountA=-2.4+0.4}{9}{%
6   \Parallel[linecolor=red,length=9](O)(A)(0,\nCountA){P1}
7   \psline[linecolor=red](0,\nCountA)(FF)
8   \psOutLine[linecolor=red,length=9](0,\nCountA)(FF){P2}
9 }
10 \psline[linecolor=blue](A)(FF)
11 \psOutLine[linecolor=blue,length=5](A)(FF){END1}
12 \rput(0,0){%
13   \lens[yBottom=-3.5,yTop=3.5,lensGlass=true,%
14     lensHeight=6.5,drawing=false,spotFi=315,lensWidth=0.5]%
15   \psline[linewidth=1pt](xLeft)(xRight)
16   \psline[length=2,linewidth=2pt,arrows=->](F')(FF)
17 }
18 \end{pspicture*}
```

Figure 18: The `\Parallel`-Macro

25 `\ABinterCD`

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. Figure 19 shows a part of figure 6. 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}`

Figure 19: `\ABinterCD`-Makro

26 `\nodeBetween`

This macro determines the coordinates of the center of a line. The syntax is:

```
\nodeBetween(A)(B){C}
```

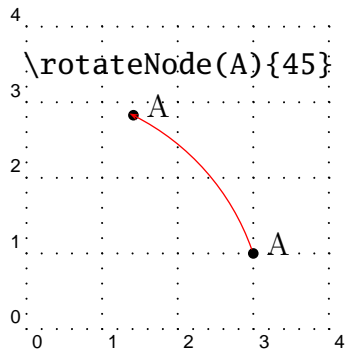
Figure 19 shows an example, where the node **M** was determined by the `\nodeBetween` macro.

27 `\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.



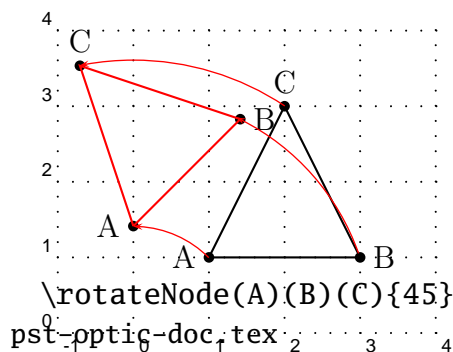
```
1 \begin{pspicture}(4,4)
2 \pnode(3,1){A}
3 \qdisk(A){2pt}\uput[20](A){A}
4 \rotateNode(A){45}
5 \qdisk(A){2pt}\uput[20](A){A}
6 \end{pspicture}
```

28 `\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.



```
1 \begin{pspicture}(-1,0)(4,4)
2 \pnode(1,1){A}
3 \pnode(3,1){B}
4 \pnode(2,3){C}
5 \qdisk(A){2pt}\uput[180](A){A}
6 \qdisk(B){2pt}\uput[0](B){B}
7 \qdisk(C){2pt}\uput[90](C){C}
```

```
\rotateNode(A)(B)(C){45}
```

```

8  \psline(A)(B)(C)(A)
9  \rotateTriangle(A)(B)(C){45}
10 \qdisk(A){2pt}\uput[180](A){A}
11 \qdisk(B){2pt}\uput[0](B){B}
12 \qdisk(C){2pt}\uput[90](C){C}
13 \psline[linecolor=red](A)(B)(C)(A)
14 \end{pspicture}

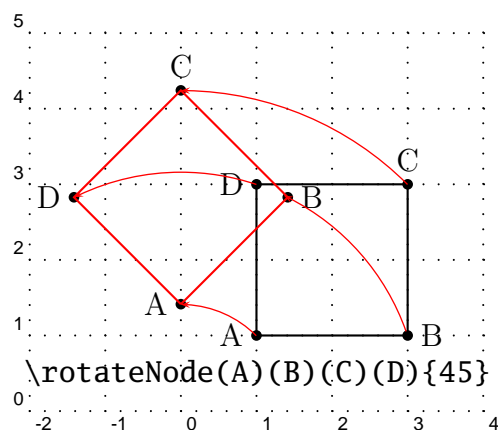
```

29 **\rotateFrame**

The syntax is

```
\rotateNode{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.



```

1 \begin{pspicture}(-2,0)(4,5)
2 \pnode(1,1){A}
3 \pnode(3,1){B}
4 \pnode(3,3){C}
5 \pnode(1,3){D}
6 \qdisk(A){2pt}\uput[180](A){A}
7 \qdisk(B){2pt}\uput[0](B){B}
8 \qdisk(C){2pt}\uput[90](C){C}
9 \qdisk(D){2pt}\uput[180](D){D}
10 \psline(A)(B)(C)(D)(A)
11 \rotateFrame(A)(B)(C)(D){45}
12 \qdisk(A){2pt}\uput[180](A){A}
13 \qdisk(B){2pt}\uput[0](B){B}
14 \qdisk(C){2pt}\uput[90](C){C}
15 \qdisk(D){2pt}\uput[180](D){D}
16 \psline[linecolor=red](A)(B)(C)(D)(A)
17 \end{pspicture}

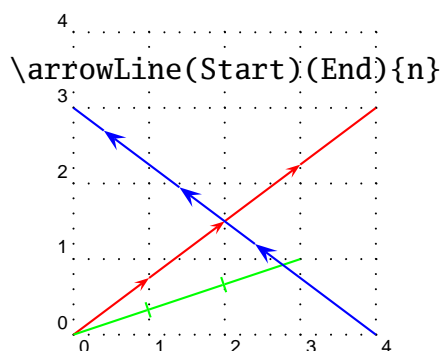
```

30 `\arrowLine`

The syntax is

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

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



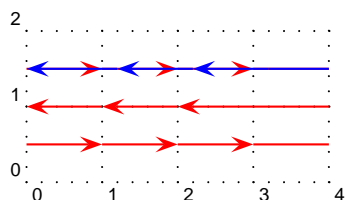
```

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

```

30.1 Options

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



```

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

```

31 Credits

Aranud Schmittbuhl |

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] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*. IWT, Vaterstetten, 1989.
- [3] Manuel Luque. *Lentilles convergentes: PST-optic v. 0.2*. <http://members.aol.com/ManuelLuque2/optique.htm>, 2001.
- [4] Herbert Voss. *PSTricks Support for pdf*. <http://PSTricks.tug.org>, 2002.
- [5] Michael Wiedmann and Peter Karp. *References for T_EX and Friends*. <http://www.miwie.org/tex-refs/>, 2003.
- [6] Timothy Van Zandt. *PSTricks - PostScript macros for Generic TeX*. <http://www.tug.org/application/PSTricks>, 1993.