The ‘pst-lens’ package
A PSTricks package for lens

Denis GIROU[*]
and
Manuel LUQUE[†]

Version 1.02 2006/06/06
Documentation revised June 6, 2006 (hv)

Abstract

This package defines a lens which can be used in various contexts to simulate the effect of a lens, using the unique macro `\PstLens`, with some customization parameters.

It is also a good example of the great power and flexibility of PSTricks, as in fact it is a very short program (it body, without considering the various customizations, is only 7 lines long!) but nevertheless powerful.

And last, it is also a good pedagogical example of how to design and program *high level graphic objects* above PSTricks own ones.

`pst-lens` uses the extended version of the keyval interface. Be sure, that you have installed the `xkeyval` package. Otherwise get it from `CTAN`.

---

[*] CNRS/IDRIS — Centre National de la Recherche Scientifique / Institut du Développement et des Ressources en Informatique Scientifique — Orsay — France — `<Denis.Girou@idris.fr>`.

[†] `<Mluque5130@aol.com>`. The original idea and the first version of the lens were from Manuel LUQUE.

Contents

1 Introduction

'pst-lens' offer a unique macro with some parameters to interact on it.

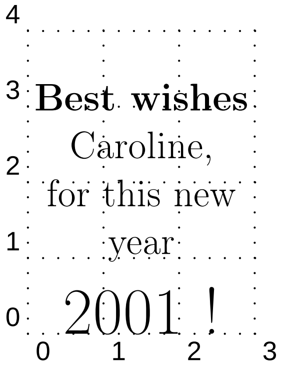
The syntax is simply: `\PstLens[optional_parameters](x,y){Object}`

`(x,y)` is a PSTricks coordinate, which as usual is taken as `(0,0)` if it is not defined.

To use the lens, we must define a `pspicture` environment, optionally draw the object and then call the `\PstLens` macro on it.

2 Usage

We will use the following textual object to illustrate our examples (note that we must define the reference point at the left bottom corner, as it is the normal behavior of PSTricks):



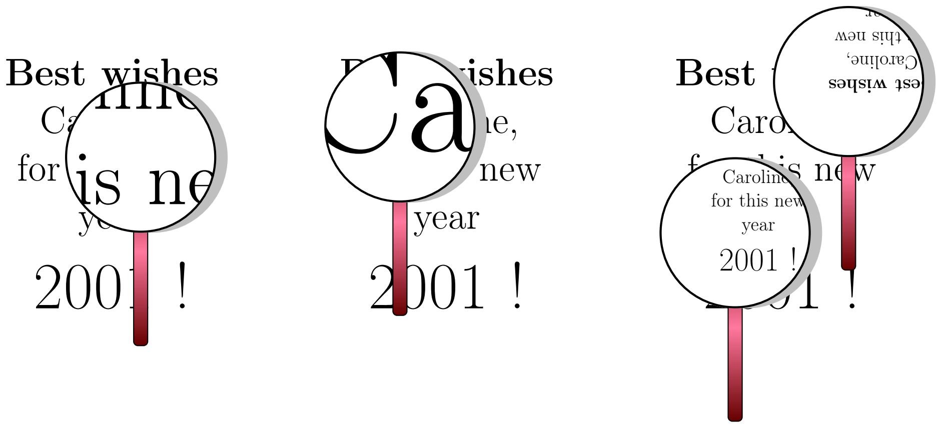
```
\def\Wishes{{%
\rput[lb](0,0){%
  \Large
  \begin{minipage}{3cm}
    \centering
    \textbf{Best wishes}\\
    Caroline,\\
    for this new year\\
    \Huge 2001 !
  \end{minipage}}}}

\Wishes
```

2.1 Parameters

There are **9** specific parameters defined to change the way the lens works:

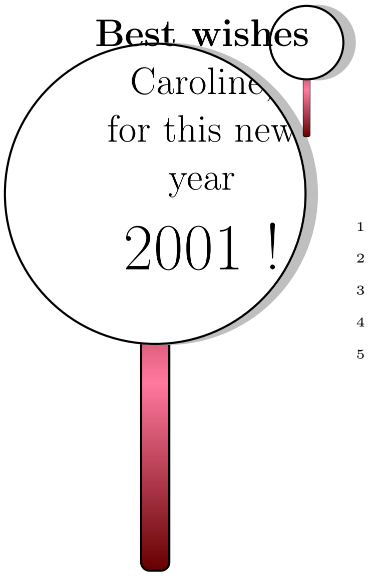
LensMagnification (real) : magnification to apply for the lens (*Default: 1* — no magnification).

```
\begin{pspicture}(0,-1.5)(3,4)
  \Wishes
  \PstLens[LensMagnification=2](2,2){\Wishes}
\end{pspicture}
\hfill
\begin{pspicture}(0,-1.5)(3,4)
  \Wishes
  \PstLens[LensMagnification=4](1,2.4){\Wishes}
\end{pspicture}
\hfill
\begin{pspicture}(0,-1.5)(3.5,4)
  \Wishes
  \PstLens[LensMagnification=0.5](1,1){\Wishes}
  \PstLens[LensMagnification=-0.5](2.5,3){\Wishes}
\end{pspicture}
```



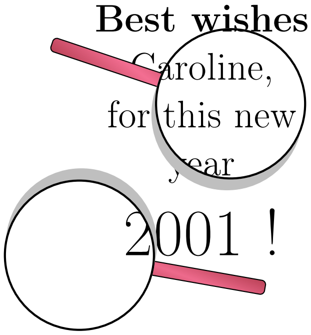
LensSize (real or length) : value of the radius of the glass of the lens (*Default: 1*).

Note that the size of the handle will change accordingly.



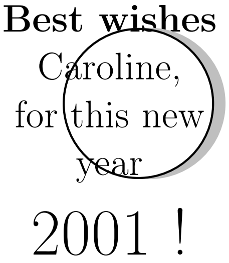
```
\begin{pspicture}(0,-4)(3,3.5)
  \Wishes
  \PstLens[LensSize=2](1,1){\Wishes}
  \PstLens[LensSize=0.5](3,3){\Wishes}
\end{pspicture}
```

LensRotation (real) : rotation angle applied to the lens (*Default: 0* — no rotation).



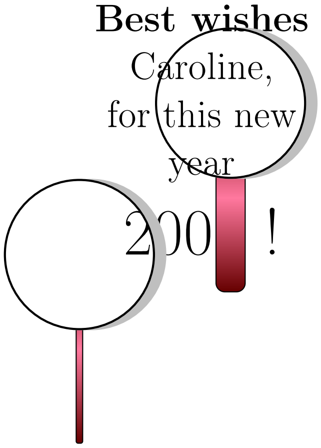
```
\begin{pspicture}(0,-1)(3,3.8)
  \Wishes
  \PstLens[LensRotation=80]{\Wishes}
  \PstLens[LensRotation=-108.5](2,2){\Wishes}
\end{pspicture}
```

LensHandle (boolean) : boolean value to choose between to draw a handle for the lens or not. (*Default: true* — handle).



```
\begin{pspicture}(3,3.5)
  \Wishes
  \PstLens[LensHandle=false](2,2){\Wishes}
\end{pspicture}
```

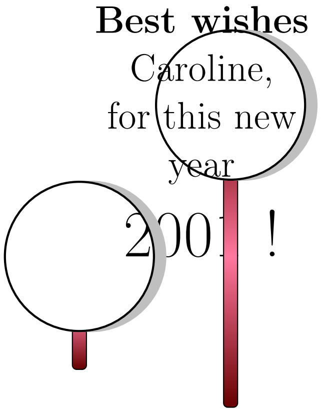
LensHandleWidth (real or length) : width of the handle (*Default: 0.2 for* `LensSize=1`).



```
\begin{pspicture}(0,-2.5)(3,3.5)
  \Wishes
  \PstLens[LensHandleWidth=0.1]{\Wishes}
  \PstLens[LensHandleWidth=4mm](2,2){\Wishes}
\end{pspicture}
```

LensHandleHeight (real or length) : height of the handle (*Default: 2.5 for* `LensSize=1`).

Take care that this length is between the *center* of the glass and the bottom of the handle.

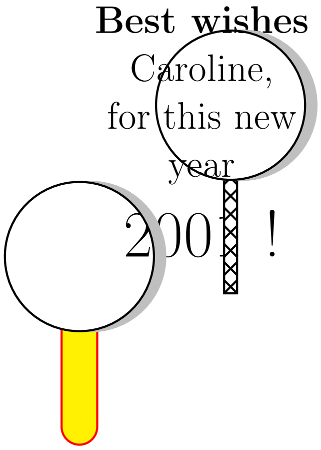


```
\begin{pspicture}(0,-2)(3,3.5)
  \Wishes
  \PstLens[LensHandleHeight=15mm]{\Wishes}
  \PstLens[LensHandleHeight=4](2,2){\Wishes}
\end{pspicture}
```

LensStyleHandle (style) : name of the PSTricks style for the handle. (*Default: LensStyleHandle*).

Its default value is:

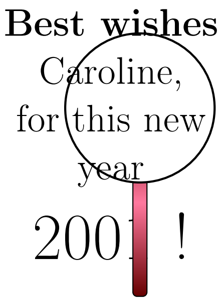
```
\newpsstyle{LensStyleHandle}{%
  fillstyle=gradient,framearc=0.6,linewidth=0.5\pslinewidth,
  gradmidpoint=0.5,gradangle=\PstLens@Rotation,
  gradbegin=Brown,gradend=Salmon}
```



```
\begin{pspicture}(0,-2.5)(3,3.5)
  \Wishes
  \newpsstyle{HandleYellow}{%
    linecolor=red,framearc=1,
    fillstyle=solid,fillcolor=yellow}
  \PstLens[LensHandleWidth=0.5,
           LensStyleHandle=HandleYellow]
          {\Wishes}
  \newpsstyle{HandleCrosshatch}{%
    fillstyle=crosshatch*,fillcolor=white}
  \PstLens[LensStyleHandle=HandleCrosshatch]
          (2,2){\Wishes}
\end{pspicture}
```

LensShadow (boolean) : boolean value to choose between to draw a shadow for the glass of the lens or not. (*Default: true* — shadow).

Note that if we redefine the `LensStyleGlass` parameter without explicitely require a shadow, there will be none even if `LensShadow` will have the `true` value.



```
\begin{pspicture}(0,-0.5)(3,3.5)
  \Wishes
  \PstLens[LensShadow=false](2,2){\Wishes}
\end{pspicture}
```

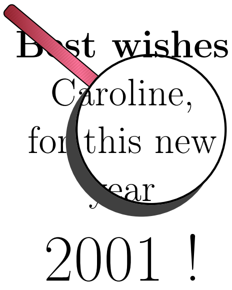
LensStyleGlass (style) : name of the PSTricks style for the glass. (*Default: LensStyleGlass*).

It allow to change the appearance of the glass, but its main utility is probably to be able to define the style of the shadow of the glass. Default definition is:

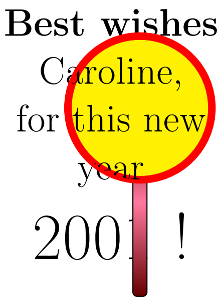
```
\newpsstyle{LensStyleGlass}{%
  fillstyle=solid,fillcolor=white,
  shadow=true,shadowcolor=lightgray,shadowsize=0.15,
  shadowangle=\PstLens@Rotation}
```

Take care that if we will use later the `LensRotation` parameter with `LensShadow` positioned, we must set the value of the `shadowangle` parameter to `\PstLens@Rotation` to have the shadow rotate accordingly.

And for better shadow effects, you must look at the 'pst-blur' package from Martin GIESE.



```
\begin{pspicture}(3,4)
  \Wishes
  \makeatletter
  \newpsstyle{DarkShadow}{%
    fillstyle=solid,fillcolor=white,
    shadow=true,shadowcolor=darkgray,
    shadowsize=0.2,
    shadowangle=\PstLens@Rotation}
  \makeatother
  \PstLens[LensRotation=230,
           LensStyleGlass=DarkShadow](2,2)
          {\Wishes}
\end{pspicture}
```

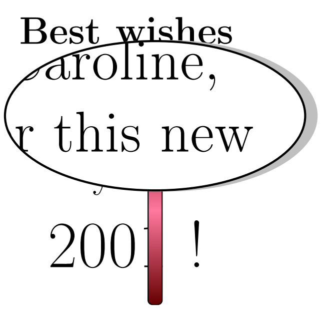


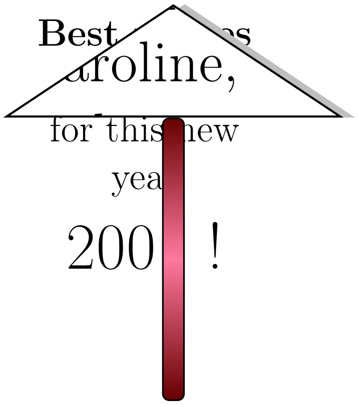
```
\begin{pspicture}(0,-0.5)(3,3.5)
  \Wishes
  \newpsstyle{YellowGlass}{%
    linecolor=red,linewidth=0.1,
    fillstyle=solid,fillcolor=yellow}
  \PstLens[LensStyleGlass=YellowGlass](2,2)
          {\Wishes}
\end{pspicture}
```

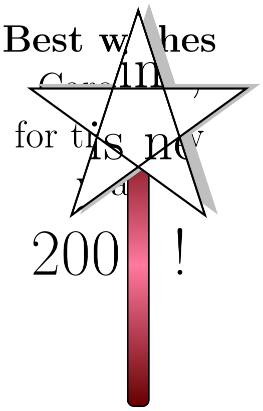
2.2 Shape of the glass

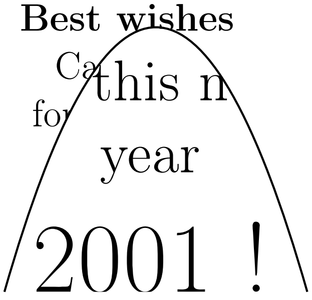
The `\PstLensShape` macro define the shape of the glass. It default value is a circle, as in real life, but we can redefine it for various effects...

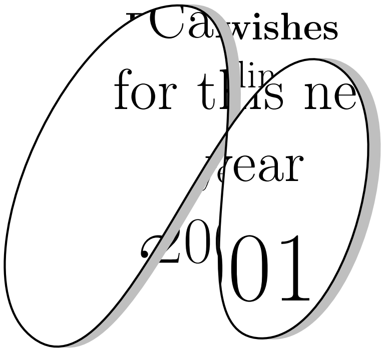
```
\psset{LensMagnification=1.5}
\begin{pspicture}(0,-1.8)(4,3.8)
  \Wishes
  \renewcommand{\PstLensShape}{\psellipse(2,1)}
  \PstLens(2,2){\Wishes}
\end{pspicture}
\hfill
\begin{pspicture}(-0.5,-1.8)(4,3.8)
  \Wishes
  \renewcommand{\PstLensShape}{\pstriangle(3,1)}
  \PstLens[LensSize=1.5](2,2){\Wishes}
\end{pspicture}
\hfill
\begin{pspicture}(0,-1.8)(3.5,3.8)
  \Wishes
  \renewcommand{\PstLensShape}{%
    \rput{18}{\pspolygon(1;0)(1;144)(1;288)(1;72)(1;216)}}
  \PstLens[LensSize=1.5](2,2){\Wishes}
\end{pspicture}

\begin{pspicture}(0,-0.5)(4,3.5)
  \renewcommand{\PstLensShape}{%
     \parabola[fillstyle=solid,fillcolor=white](-1,-1.5)(1,2)}
  \Wishes
  \PstLens[LensShadow=false,LensHandle=false](1,1){\Wishes}
\end{pspicture}
\hfill
\begin{pspicture}(-1.5,-1)(3.5,3.5)
  \renewcommand{\PstLensShape}{%
    \psccurve(-1,-1)(0,1.2)(0.5,-1)(1,0.8)}
  \Wishes
  \PstLens[LensSize=2,LensHandle=false](1,1){\Wishes}
\end{pspicture}
```











2.3 Examples

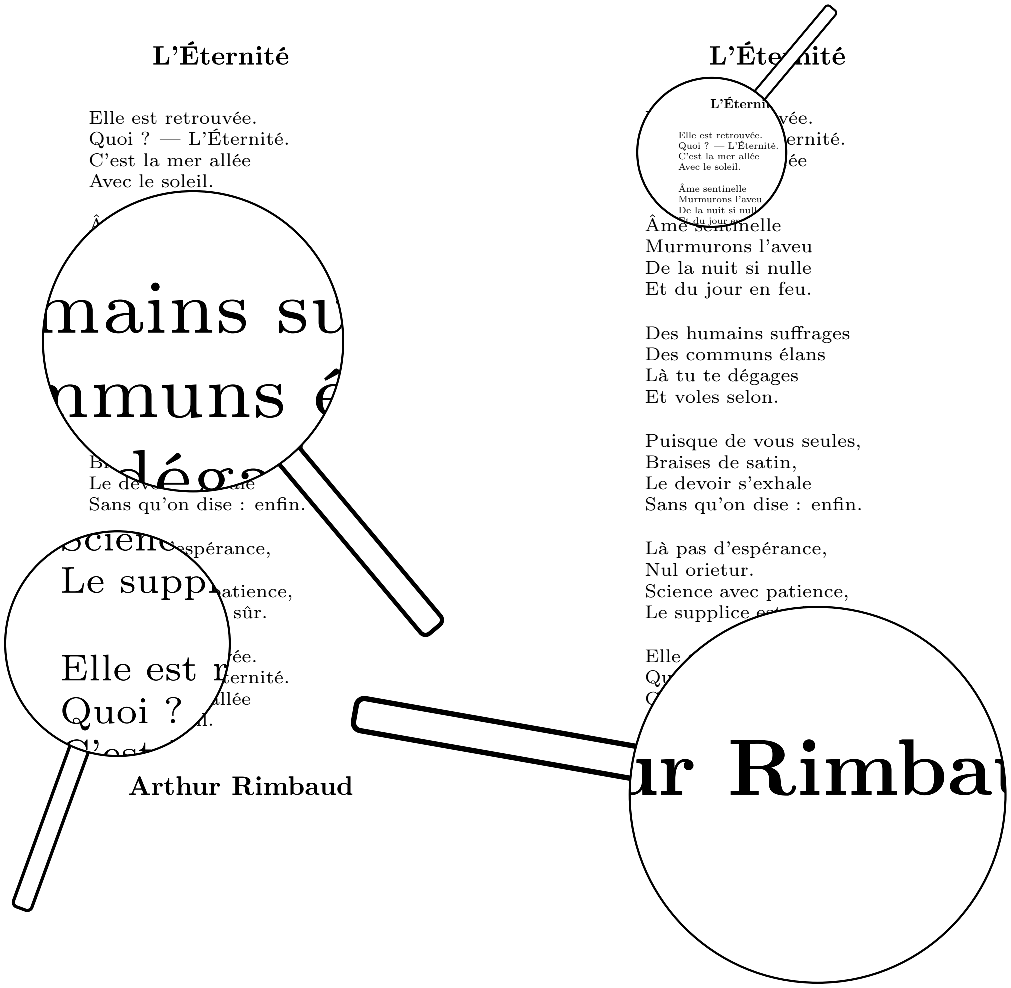
We can use the lens for all textual objects and for all PSTricks graphic objects (we use here some versions of tilings and fractals, but only basic ones to avoid requiring too much memory from old TEX systems, to compile the file).

And specially take care to explicitely position the reference point at the left bottom corner and to compute the correct dimensions for the `pspicture` environment (in our examples, we choose most of the time to include the lens inside the bounding boxes, but we can choose to define them just for the objects).

```
\def\TheEternity{{%
\rput[lb](0,0){%
  \scriptsize
  \begin{minipage}{3.5cm}
    \centerline{\normalsize\textbf{L'\'Eternit\'e}}
...
```

```
\def\TruchetTiling#1#2{{%
\rput[lb](0,0){%
...
\def\PstSierpinskiTriangle#1{{%
\rput[lb](0,0){%
...
\def\PstVonKochCurve#1{{%
\rput[lb](0,0){%
...
```

```
\newpsstyle{SimpleGlass}{fillstyle=solid,fillcolor=white}
\newpsstyle{SimpleHandle}{fillstyle=solid,fillcolor=white,
                          framearc=0.5}
\psset{LensStyleGlass=SimpleGlass,LensStyleHandle=SimpleHandle}

\begin{pspicture}(-1,-2.5)(5,10.5)
  \TheEternity
  \PstLens[LensSize=2,LensMagnification=4,LensRotation=40]
          (1.5,6){\TheEternity}
  \PstLens[LensSize=1.5,LensMagnification=2,LensRotation=-20]
          (0.5,2){\TheEternity}
\end{pspicture}
\hfill
\begin{pspicture}(-2,-2.5)(4,10.5)
  \TheEternity
  \PstLens[LensMagnification=0.5,LensRotation=140]
          (1,8.5){\TheEternity}
  \PstLens[LensSize=2.5,LensMagnification=3,LensRotation=-100]
          (2.4,0){\TheEternity}
\end{pspicture}
```



L'Éternité

Elle est retrouvée.
Quoi ? — L'Éternité.
C'est la mer allée
Avec le soleil.

mains su
mmuns é
dégag

Le devoir s'exhale
Sans qu'on dise : enfin.

Scienc
Le suppl

Elle est r
Quoi ?
C'est

Arthur Rimbaud

L'Éternité

Elle est retrouvée.
Quoi ? — L'Éternité.
C'est la mer allée
Avec le soleil.

Âme sentinelle
Murmurons l'aveu
De la nuit si nulle
Et du jour en feu.

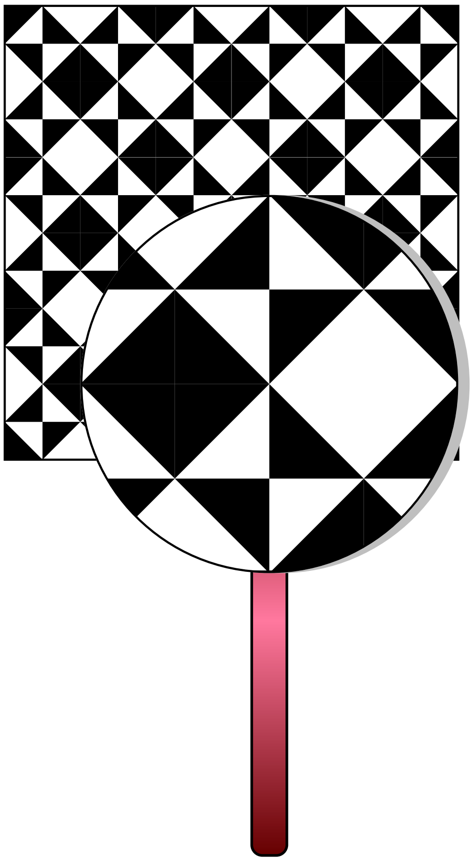
Des humains suffrages
Des communs élans
Là tu te dégages
Et voles selon.

Puisque de vous seules,
Braises de satin,
Le devoir s'exhale
Sans qu'on dise : enfin.

Là pas d'espérance,
Nul orietur.
Science avec patience,
Le supplice

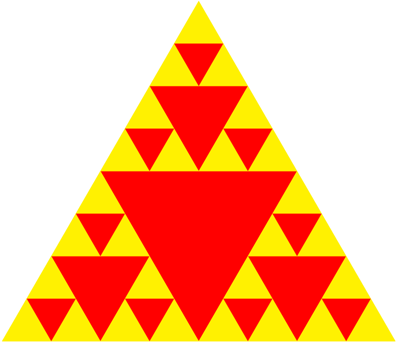
ur Rimbau

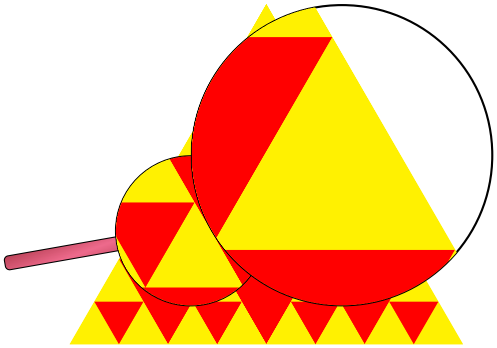
```
\begin{pspicture}(0,-6)(6,6)
  \TruchetTiling
  \PstLens[LensSize=2.5,LensMagnification=2.5](3.5,1)
          {\TruchetTiling}
\end{pspicture}
```

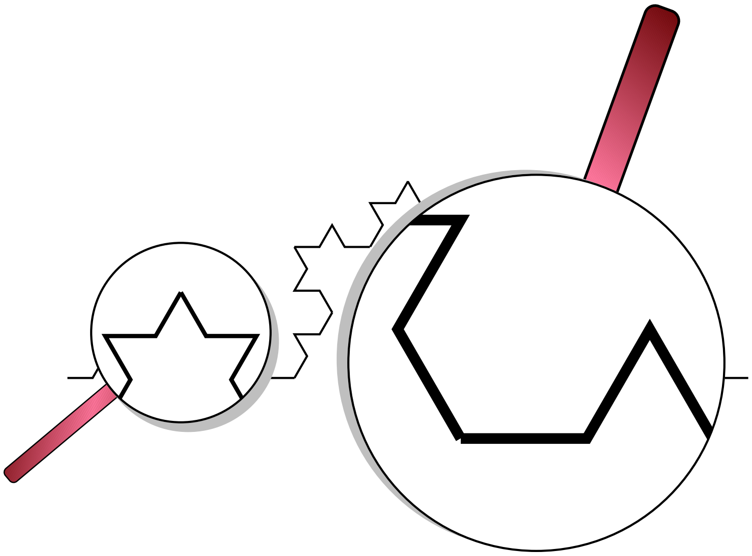


```
\newcommand{\PstSierpinskiInternalColor}{red}
\newcommand{\PstSierpinskiExternalColor}{yellow}

% The Sierpinski triangle is in a unit circle of radius 1,
% so we must define the "pspicture" accordingly: (-3,-2)(3,3)
\begin{pspicture}(-3,-2)(3,3)
  \PstSierpinskiTriangle{3}
\end{pspicture}
\hfill
\begin{pspicture}(-3,-2)(3,3)
  \PstSierpinskiTriangle{3}
  \psset{LensShadow=false}
  \PstLens[LensMagnification=2,LensRotation=-80](-1,0)
          {\PstSierpinskiTriangle{3}}
  \PstLens[LensSize=2,LensMagnification=5,LensRotation=100,
           LensHandle=false](1,1){\PstSierpinskiTriangle{3}}
```

```
\end{pspicture}

\begin{pspicture}(-1,-2)(11,5)
  \PstVonKochCurve{3}
  \PstLens[LensSize=1.2,LensMagnification=2,LensRotation=-50]
          (1.5,0.6){\PstVonKochCurve{3}}
  \PstLens[LensSize=2.5,LensMagnification=5,LensRotation=160,
           LensHandleHeight=2](6.2,0.2){\PstVonKochCurve{3}}
\end{pspicture}
```



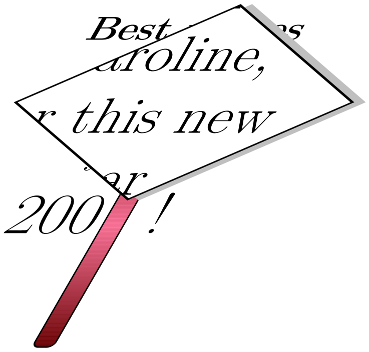


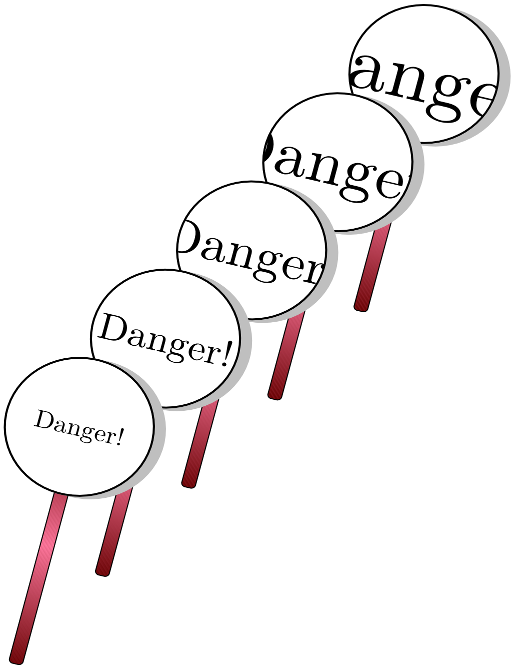


Of course, as for all PSTricks objects, we can apply to them some transformations. For instance, we can project them in the 3 dimensional space, with the general `\ThreeDput` macro or the simple `\pstilt` one.

```
\psset{LensMagnification=1.5}
\begin{pspicture}(0.8,-1.5)(5.3,3)
```

```
  \renewcommand{\PstLensShape}{\psdiamond(1.5,1)}
  \pstilt{60}{%
    \Wishes
    \PstLens[LensSize=1.5](2,2){\Wishes}}
\end{pspicture}
\hfill
\begin{pspicture}(-3,-0.5)(3.5,8)
  \psset{viewpoint=0.5 -2 5,LensHandleHeight=3.5}
  \multido{\nPosX=0+-0.8,\nPosY=8+-1.5,\nMag=3+-0.5}{5}{%
    \ThreeDput(\nPosX,\nPosY,0){%
      \PstLens[LensMagnification=\nMag](0.6,0.2)
              {\rput[lb](0,0){Danger!}}}}
\end{pspicture}
```

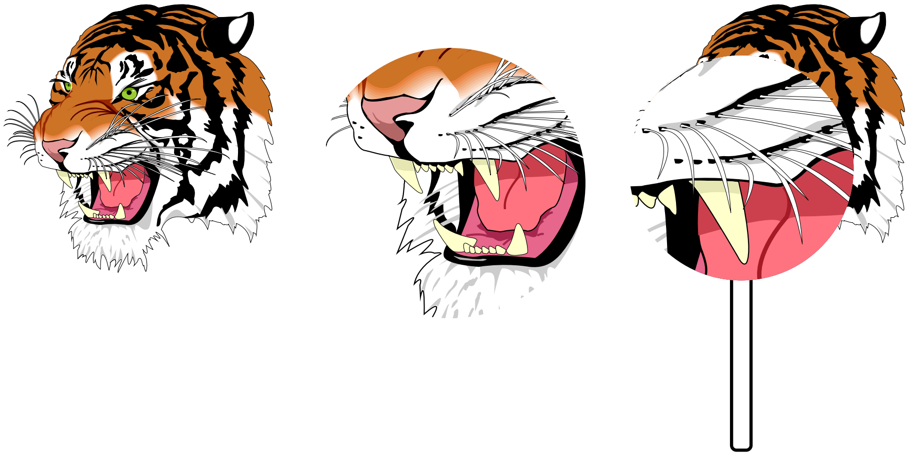




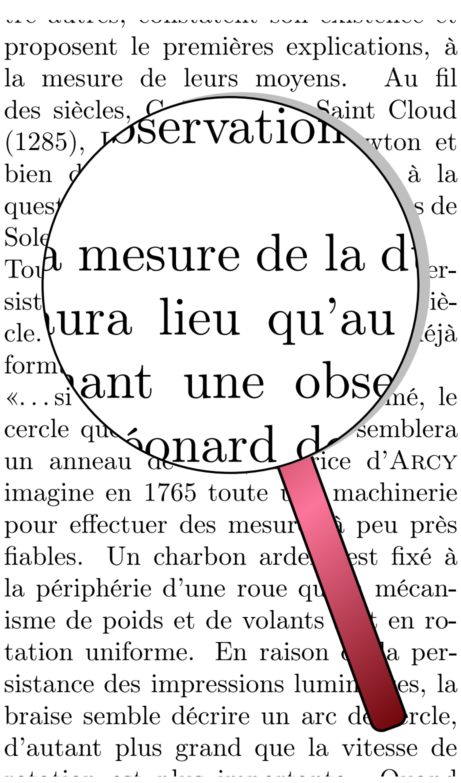
And we can also use the lens on non PSTricks graphics, as external images.

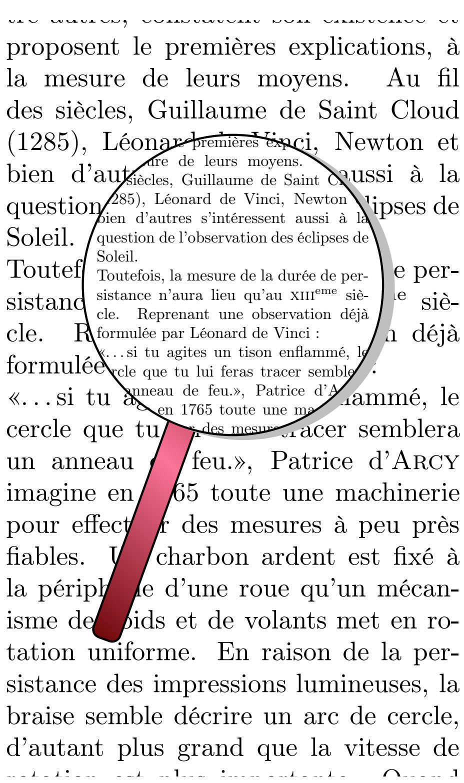
```
\newcommand{\tigerHead}{%
  \rput[lb](0,0){%
    \includegraphics[width=4cm,height=5cm]{tiger}}}
\newpsstyle{SimpleGlass}{linestyle=none}
\psset{LensStyleGlass=SimpleGlass}

\begin{pspicture}(0,-1)(4,5)
```

```
  \tigerHead
\end{pspicture}
\hfill
\begin{pspicture}(-0.5,-1)(3,5)
  \PstLens[LensHandle=false,LensSize=1.8,LensMagnification=2]
          (1.2,2.3){\tigerHead}
\end{pspicture}
\hfill
\newpsstyle{SimpleHandle}{fillstyle=solid,fillcolor=white,
                          framearc=0.5}
\psset{LensStyleHandle=SimpleHandle}
\begin{pspicture}(0,-1)(4,5)
  \tigerHead
  \PstLens[LensSize=1.5,LensMagnification=4]
          (1.5,2.5){\tigerHead}
\end{pspicture}
```



```
\def\Persistance{{%
\rput(0,0){%
\begin{minipage}{6cm}
 \centerline{\normalsize\textbf{La persistance rétinienne}}
 \vspace{5mm}
La persistance des impressions lumineuses, ou maintien de la
sensation lumineuse après que l'excitation ait disparue, est
connue depuis la plus haute antiquité. \textsc{Aristote}
(Sur le Songes) et \textsc{Lucrèce} (De Natura Rerum),
entre autres, constatent son existence et proposent le
premières explications, à la mesure de leurs moyens.
Au fil des siècles, Guillaume de Saint Cloud (1285),
Léonard de Vinci, Newton et bien d'autres s'intéressent aussi
à la question de l'observation des éclipses de Soleil.

Toutefois, la mesure de la durée de persistance n'aura lieu qu'au
\textsc{xiii}\textsuperscript{eme} siècle. Reprenant une
 observation déjà formulée par Léonard de Vinci :

<<\ldots si tu agites un tison enflammé, le cercle que tu lui feras
tracer semblera un anneau de feu.>>, Patrice d'\textsc{Arcy}
imagine en 1765 toute une machinerie pour effectuer des mesures à
peu près fiables. Un charbon ardent est fixé à la périphérie d'une
roue qu'un mécanisme de poids et de volants met en rotation
uniforme. En raison de la persistance des impressions lumineuses,
la braise semble décrire un arc de cercle, d'autant plus grand que
la vitesse de rotation est plus importante. Quand la durée d'un
tour est égale à celle de la persistance de la sensation lumineuse,
la trace décrit un tour complet. À la suite de nombreuses
expériences, d'\textsc{Arcy} aboutit à la valeur de 8 tierces, à
peu près 130 millisecondes.
 \flushright{\normalsize\textbf{Miche HENRY}}
\end{minipage}}}}

\begin{pspicture}*(-3,-5)(3,5)
 \Persistance
 \PstLens[LensSize=2.5,LensMagnification=2,LensRotation=20]%
    (0,1.5){\Persistance}
\end{pspicture}\hfill
\begin{pspicture}*(-3,-5)(3,5)
 \Persistance
 \PstLens[LensSize=2,LensMagnification=0.6,LensRotation=-20]%
    (0,1.5){\Persistance}
\end{pspicture}
```





3 Driver file

The next bit of code contains the documentation driver file for TEX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the `docstrip` program.

```
⟨*driver⟩
\documentclass{ltxdoc}
\GetFileInfo{pst-lens.dtx}
\usepackage[T1]{fontenc}
\usepackage{fancyvrb}
\usepackage{graphicx}
\usepackage{ifthen}
\usepackage{multido}
\usepackage{pstricks}
\usepackage{pst-lens}
\let\pstLensFileVersion\fileversion
\let\pstLensFileDate\filedate
\usepackage{pst-3d}
\AtBeginDocument{
%  \OnlyDescription % comment out for implementation details
  \EnableCrossrefs
  \RecordChanges
  \CodelineIndex
}
\AtEndDocument{
  \PrintChanges
  \PrintIndex
}
\hbadness=7000            % Over and under full box warnings
\hfuzz=3pt
\begin{document}
  \DocInput{pst-lens.dtx}
\end{document}
⟨/driver⟩
```

4 ‘pst-lens’ LaTeX wrapper

```
⟨*latex − wrapper⟩
\RequirePackage{pstricks}
\ProvidesPackage{pst-lens}[2005/09/02 package wrapper for
  pst-lens.tex (hv)]
\input{pst-lens.tex}
\ProvidesFile{pst-lens.tex}
  [\filedate\space v\fileversion\space ‘PST-lens’ (hv)]
⟨/latex − wrapper⟩
```

5 ‘pst-lens’ code

```
⟨*pst − lens⟩
```

5.1 Preambule

Who we are.

```
\csname PSTLensLoaded\endcsname
\let\PSTLensLoaded\endinput
```

Require the PSTricks and ‘pst-grad’ packages.

```
\ifx\PSTricksLoaded\endinput\else\input pstricks.tex\fi
\ifx\GradientLoaded\endinput\else\input pst-grad.tex\fi
\ifx\PSTXKeyLoaded\endinput\else\input pst-xkey \fi % (hv 2005-09-03)
```

Catcodes changes.

```
\edef\PstAtCode{\the\catcode‘\@}
\catcode‘\@=11\relax
\pst@addfams{pst-lens}
\def\fileversion{1.02}
\def\filedate{2006/06/06}
\message{‘PST-Lens’ v\fileversion, \filedate\space
        (Denis Girou and Manuel Luque)}
```

5.2 Definition of the parameters

`LensHandle` will define if we will draw an handle to the lens or not. It is a *boolean* value.

```
\newif\ifPstLens@Handle
\define@key[psset]{pst-lens}{LensHandle}[true]{%
  \@nameuse{PstLens@Handle#1}}
```

`LensStyleHandle` is the name of the PSTricks style to draw the handle of the lens.

```
\define@key[psset]{pst-lens}{LensStyleHandle}{%
  \def\PstLens@StyleHandle{#1}}
```

`LensHandleWidth` will be the size of the lens. This is a *real* or *length* value, as all PSTrisks dimensions, but as we will have to make computations with it, we store it in a dimension register.

```
\newdimen\PstLens@HandleWidth
\define@key[psset]{pst-lens}{LensHandleWidth}{%
  \pssetlength{\PstLens@HandleWidth}{#1}}
```

`LensHandleHeight` will be the size of the lens. This is a *real* or *length* value, as all PSTrisks dimensions, but as we will have not to make computations with it, we store it in a simple macro.

```
\define@key[psset]{pst-lens}{LensHandleHeight}{%
  \def\PstLens@HandleHeight{#1}}
```

`LensShadow` will define if we will draw a shadow to the glass of the lens or not. It is a *boolean* value.

```
\newif\ifPstLens@Shadow
\define@key[psset]{pst-lens}{LensShadow}[true]{%
  \@nameuse{PstLens@Shadow#1}}
```

`LensStyleGlass` is the name of the PSTricks style to draw the glass.

```
\define@key[psset]{pst-lens}{LensStyleGlass}{%
  \def\PstLens@StyleGlass{#1}}
```

`LensSize` will be the size of the lens. This is a *real* or *length* value, as all PSTrisks dimensions, but as we will have not to make computations with it, we store it in a simple macro.

```
\define@key[psset]{pst-lens}{LensSize}{%
  \def\PstLens@Size{#1}}
```

`LensMagnification` will be the magnification to apply to the lens. This is a *real* or *length* value, but as we will have not to make computations with it, we strore it in a simple macro.

```
\define@key[psset]{pst-lens}{LensMagnification}{%
  \def\PstLens@Magnification{#1}}
```

`LensRotation` will be the rotation angle to apply to the lens. It is a *real* value used as an *angle*.

```
\define@key[psset]{pst-lens}{LensRotation}{%
  \def\PstLens@Rotation{#1}}
```

Next, we set the default values for all these new parameters. We choose to have an handle of width 0.2 unit and height of 2.5 unit, LensStyleHandle as style for the handle, a shadow to the glass, LensStyleGlass as style for it, no rotation, a size equal to 1 unit and no magnification (so of value 1).

```
\psset[pst-lens]{%
  LensHandle=true,LensHandleWidth=0.2,LensHandleHeight=2.5,
  LensStyleHandle=LensStyleHandle,
  LensShadow=true,LensStyleGlass=LensStyleGlass,
  LensRotation=0,LensSize=1,LensMagnification=1}
```

We define also the default style for the handle.

```
\newcmykcolor{Brown}{0 0.81 1 0.6}
\newcmykcolor{Salmon}{0 0.53 0.38 0}
\newpsstyle{LensStyleHandle}{%
  fillstyle=gradient,framearc=0.6,linewidth=0.5\pslinewidth,
  gradmidpoint=0.5,gradangle=\PstLens@Rotation,gradbegin=Brown,gradend=Salmon}
```

And the default style for the glass (we only define a shaow for it).

```
\newpsstyle{LensStyleGlass}{%
  fillstyle=solid,fillcolor=white,
  shadow=true,shadowcolor=lightgray,shadowsize=0.15,
  shadowangle=\PstLens@Rotation}
```

Then we define the default shape for the lens (a circle of radius 1 and center (0,0)).

```
\def\PstLensShape{\pscircle{1}}
```

5.3 Main macro

The general `\PstLens` macro to draw lens.

`\PstLens`

```
\def\PstLens{\@ifnextchar[{\PstLens@i}{\PstLens@i[]}}
```

We first check if the coordinate is given, and if not we choose as usual (0,0) as default one.

`\PstLens@i`

```
\def\PstLens@i[#1]{\@ifnextchar({\PstLens@ii[#1]}{\PstLens@ii[#1](0,0)}}
```

Then we define the auxiliary macro which will handle the parameters if some are used. Note also the usage of the double `{{` to have only changes of parameter values for this specific object, as redefinitions of them must be local.

\PstLens@ii

```
\def\PstLens@ii[#1](#2,#3)#4{{%
```

After that, we can set the values of local parameters if defined.

```
\psset{#1}%
```

Now, we can start the *real* code. First, we must be able to use PostScript expressions as coordinates.

```
\SpecialCoor
```

Then, if the handle is not suppressed, we position it at the required coordinate, with it specified rotation.

```
\rput{\PstLens@Rotation}(#2,#3){%
  \ifPstLens@Handle
    \psscalebox{\PstLens@Size}{%
      \divide\PstLens@HandleWidth\tw@
      \psframe[style=\PstLens@StyleHandle]
              (-\PstLens@HandleWidth,0)
              (\PstLens@HandleWidth,-\PstLens@HandleHeight)}%
  \fi}
```

Now, we can draw the object at the specified position, but surimposing on it a shape which will represent the glass of the lens, and we use the powerful clipping mechanism to eliminate the parts of the object not inside this shape.

```
\psclip{{\psset{style=\PstLens@StyleGlass}
         \ifPstLens@Shadow
         \else
           \psset{shadow=false}
         \fi
         \rput(#2,#3){\psset{unit=\PstLens@Size}\PstLensShape}}}
```

Then we draw again the object, but with the specified magnification. It require also to recompute the coordinates where to put the object, according to the magnification.

```
  \rput(! 1 \PstLens@Magnification\space sub #2\space mul
          1 \PstLens@Magnification\space sub #3\space mul){%
    \psscalebox{\PstLens@Magnification}{#4}}
```

To finish we close the clipping mechanism and the `\PstLens@ii` macro.

```
\endpsclip}}
```

5.4 Closing

Catcodes restoration.

```
\catcode`\@=\PstAtCode\relax
⟨/pst − lens⟩
```

Change History

v1.00
- General: First public release. (dg) 1

v1.01
- General: using the extended pst-xkey instead of pst-key. (hv) . . 1

v1.02
- General: using psscalebox instead of scalebox. (hv) 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.