GraphicxSP: Re-using EPS files

D. P. Story
Email: dpstory@acrotex.net & storyd@owc.edu

processed December 1, 2019

Contents

1 ⟨∗package⟩

1 Introduction

GraphicxSP is a patch into the graphicx package so that users of dvips and dvipsone, using distiller, can insert and re-use `.eps` figures.

2 Package Options

`preview`
`dvipsone`
`dvips`
`showembeds`
`!showembeds`

This package recognizes three options: driver names `dvips` (the default), `dvipsone` (old YandY TEX) and a `preview`. `dvipsone`, using distiller, can insert and re-use `.eps` figures.

```
\@ifundefined{ifpreview}{\newif\ifpreview\previewfalse}{}
```

(2017/03/12) Added two convenience commands.

```
\providecommand{\previewOn}{\previewtrue}
\providecommand{\previewOff}{\previewfalse}
\DeclareOption{preview}{\previewtrue}
\DeclareOption{dvipsone}{\def\gxsp@drivernum{0}}
\DeclareOption{dvips}{\def\gxsp@drivernum{1}}
\DeclareOption{showembeds}{\let\gxsp@showembeds=0}
\DeclareOption{!showembeds}{\let\gxsp@showembeds=1}
\let\gxsp@showembeds=1
\def\gxsp@drivernum{1}
```

`draft`
`final`
`shownonames`
`!shownonames`

The `draft` mode passes `draft` on to graphicx. The images appear as rectangles, with the name of the image. The `shownonames` option removes the name inside the rectangle.

```
\DeclareOption{draft}{\spxGin@drafttrue
    \PassOptionsToPackage{draft}{graphicx}}
\DeclareOption{!draft}{}
\DeclareOption{final}{\spxGin@draftfalse
    \PassOptionsToPackage{final}{graphicx}}
\DeclareOption{shownonames}{\@spx@shownameindraftfalse}
\DeclareOption{!shownonames}{\@spx@shownameindrafttrue}
\newif\if@spx@shownameindraft \@spx@shownameindrafttrue
\newif\ifspxGin@draft \spxGin@draftfalse
```

```
\InputIfFileExists{graphics.cfg}{}{}
\ProcessOptions
\@ifundefined{eq@driver@name}{}{%
```

As a point of personal convenience, if `\eq@driver@name`, which is defined in web, and the name is `dvipsone`, we'll override the default of `dvips`.

```
\def\DVIPSONE{dvipsone}\ifx\eq@driver@name\DVIPSONE
\def\gxsp@drivernum{0}\fi}
```

3 Package Requirements

We minimally require the graphicx package, which we patch into, and the eso-pic, package, which, in turn, requires the everyshi package.

```
\RequirePackage{graphicx}
\ifspxGin@draft\Gin@drafttrue\fi
\RequirePackage{eso-pic}
\RequirePackage{verbatim}
```

4 PostScript and Driver Dependent Definitions

Hyperref is not required, but if present, we'll use its code, otherwise, we use code from hyperref to hide in-line images from GhostScript's view.

```
\def\grcxsp@hideEPS{\AtBeginDvi{\special{!%
/product where{%
pop
product(Distiller)search{%
    pop pop pop
    userdict
    /?pdfmark /exec load put%
}{%
pop
userdict
    begin
    /?pdfmark /pop load def
    end
}ifelse%
}if%
}}}
\@ifpackageloaded{hyperref}{\let\grcxsp@hideEPS\relax}{\grcxsp@hideEPS}
```

We use either dvips or dvipsone as the driver, in both cases the following is the special we shall use.

```
\def\gxsp@psMrk{[%]
  \space}
\def\gxsp@literalps@out#1{\special{ps:#1}}
```

The following are driver dependent definitions. We begin with `dvips`.

Dvips driver.

```
\ifnum\gxsp@drivernum=1\relax
```

When the driver is `dvips`, we define some postscript procedures to making conversions between TEX and PDF.

```
\special{!userdict begin
   /TeXtoPDF {65536 div DVImag mul} def         % sp to pts
   /PDFtoDvips {72.27 div Resolution mul} def   % points to dots
   /PDFtoVDvips {72.27 div VResolution mul} def % points to dots
   /DvipstoPDF {72.27 mul Resolution div} def   % dots to points
   /HTeXtoDvips {TeXtoPDF PDFtoDvips} def       % sp to dots
   /VTeXtoDvips {TeXtoPDF PDFtoVDvips} def end} % sp to dots
```

The `cstr` is used to calculate the lower left corner of the bounding box of an EPS file for `dvips`.

```
\special{!userdict begin /cstr {currentpoint translate
   1 PDFtoDvips DVImag mul -1 PDFtoDvips DVImag mul scale}def end}
\def\gxsp@setPSCoor{cstr }
\def\b@grxsp@Literal{userdict begin}
\def\e@grxsp@Literal{end}
\else
```

Dvipsone driver. The following is code special to the dvipsone driver.

The `undsclx` is used to calculate the lower left corner of the the bounding box of an EPS file for `dvipsone`.

```
\def\gxsp@setPSCoor{undsclx }
\let\b@grxsp@Literal\@empty
\let\e@grxsp@Literal\@empty
\fi
```

These are procedures that support the dynamic naming of `/_objdef`. Distiller crashes if any symbolic reference name is not unique. So we must protect distiller. `grcxspObjDef` takes a single argument on the operand stack

```
(<name>) grcxspObjDef
```

and leaves at the top of the stack `(<name>)graphicxspCnt-currentpage)`

```
\special{!\b@grxsp@Literal
    /currentpage 0 def
    /graphicxspCnt 0 def
    /graphicxspStr 10 string def
    /graphicxspMergeStr {2 copy length exch length add string dup dup
    4 3 roll 4 index length exch putinterval 3 1 roll exch
    0 exch putinterval} def
    /grcxspObjDef {
        /graphicxspCnt graphicxspCnt 1 add def
        currentpage graphicxspStr cvs graphicxspMergeStr
        (-) graphicxspMergeStr
        graphicxspCnt graphicxspStr cvs
        graphicxspMergeStr
%       dup (Creating _objdef ) exch (\string\n) graphicxspMergeStr
%       graphicxspMergeStr print flush
    } def \e@grxsp@Literal
}
```

5 Messing with eso-pic

One of the problems was to embed EPS files within a **BP**/**EP** operator pair. The solutions was to use eso-pic, place each graphic at the lower left corner of the page. We define a new “Hook” for eso-pic and attach it to `\@ShipoutPicture`.

```
\def\ESO@AeBip@Hook{}
\newcommand{\AddToEmbeddedEPSs}{\g@addto@macro\ESO@AeBip@Hook}
```

We redefine `\@ShipoutPicture` command of eso-pic so that the embedded figures are placed before `ESO@HookI` and `ESO@HookII`, for the case that someone wants to use placed pictures for a background, the file must be embedded before they can be inserted.

```
\@ifundefined{@ShipoutPicture}{%
```

The new version of eso-pic does not define `\@ShipoutPicture`, so we use some of the new code.

```
    \ESO@isMEMOIR{%
```

```
      \AtBeginShipout{%
        \@tempdima=-\trimedge
        \advance\@tempdima-\paperwidth
        \advance\@tempdima\stockwidth
        \if@twoside\ifodd\c@page\else
          \advance\@tempdima2\trimedge
          \advance\@tempdima\paperwidth
          \advance\@tempdima-\stockwidth
        \fi\fi
        \@tempdimb=\ESO@yoffsetI
        \advance\@tempdimb-\trimtop
        \nointerlineskip
        \AtBeginShipoutUpperLeft{%
          \put(\LenToUnit{\@tempdima},\LenToUnit{\@tempdimb}){%
            \ESO@HookIII\ESO@HookI\ESO@HookII
            \global\let\ESO@HookII\@empty
          }%
        }%
      }
    }{%
      \AtBeginShipout{%
        \nointerlineskip
        \AtBeginShipoutUpperLeft{%
          \put(0,\LenToUnit{\ESO@yoffsetI}){%
            \ESO@HookIII\ESO@AeBip@Hook\ESO@HookI\ESO@HookII% dps
            \global\let\ESO@HookII\@empty
            \global\let\ESO@AeBip@Hook\@empty% dps
          }%
        }%
      }
    }
}{%
```

If `\@ShipoutPicture` is defined, we use the old code.

```
    \renewcommand{\@ShipoutPicture}{%
      \bgroup
        \@tempswafalse%
        \ifx\ESO@HookI\@empty\else\@tempswatrue\fi%
        \ifx\ESO@HookII\@empty\else\@tempswatrue\fi%
        \ifx\ESO@HookIII\@empty\else\@tempswatrue\fi%
        \ifx\ESO@AeBip@Hook\@empty\else\@tempswatrue\fi%dps(08/16/07)
         \if@tempswa%
          \@tempdima=1in\@tempdimb=-\@tempdima%
          \advance\@tempdimb\ESO@yoffsetI%
          \ESO@isMEMOIR{%
            \advance\@tempdima\trimedge%
            \advance\@tempdima\paperwidth%
            \advance\@tempdima-\stockwidth%
            \if@twoside\ifodd\c@page\else%
              \advance\@tempdima-2\trimedge%
              \advance\@tempdima-\paperwidth%
              \advance\@tempdima\stockwidth%
            \fi\fi%
            \advance\@tempdimb\trimtop}%
          \unitlength=1pt%
          \global\setbox\@cclv\vbox{%
            \vbox{\let\protect\relax
              \pictur@(0,0)(\strip@pt\@tempdima,\strip@pt\@tempdimb)%
                \ESO@HookIII\ESO@AeBip@Hook\ESO@HookI\ESO@HookII%dps
                \global\let\ESO@HookII\@empty%
                \global\let\ESO@AeBip@Hook\@empty%              %dps
              \endpicture}%
              \nointerlineskip%
            \box\@cclv}%
        \fi
      \egroup
    }
}
```

```
\AddToShipoutPicture{\special{ps: /currentpage \thepage\space def}}
```

6 Useful Supporting Commands

Some standard code that I use in AeB to wrote verbatim tex to a file.

```
\def\verbatimwrite{\@bsphack
    \let\do\@makeother\dospecials
    \catcode`\^^M\active \catcode`\^^I=12
    \def\verbatim@processline{%
        \immediate\write\verbatim@out
        {\the\verbatim@line}}%
    \verbatim@start
}
\def\endverbatimwrite{\@esphack}
\def\gxsp@IWVO{\immediate\write\verbatim@out}
\def\x@namedef#1{\expandafter\xdef\csname #1\endcsname}
\def\e@namedef#1{\expandafter\edef\csname #1\endcsname}
```

Below is a counter to ensure each name is unique. It is used in `\Ginclude@eps@SP`.

```
\newcount\grxsp@cnt \grxsp@cnt=0
```

7 The Main Section

In this section we define two commands for the user, `\embedEPS` and `\insertEPS`, defined additional keys for the graphicx package, and consequently, hook into the `\includegraphics` command.

Some helper commands to save the dimensions of the pictures as they are embedded using `\embedEPS`.

```
\def\grcxsp@setPictureDimen#1#2#3#4#5{%
```

```
    \x@namedef{#1Gin@llx}{#2}\x@namedef{#1Gin@lly}{#3}%
    \x@namedef{#1Gin@urx}{#4}\x@namedef{#1Gin@ury}{#5}%
    \x@namedef{#1BBox}{#2 #3 #4 #5}%
    \begingroup
```

Calculate the width and height of the EPS. If the lower-left corner is not (0,0), results may not be predictable.

```
        \@tempdima=#4bp
        \advance\@tempdima-#2bp
        \@tempdima=.99626\@tempdima
        \x@namedef{#1widthOf}{\strip@pt\@tempdima}%
        \@tempdima=#5bp
        \advance\@tempdima-#3bp
        \@tempdima=.99626\@tempdima
        \x@namedef{#1heightOf}{\strip@pt\@tempdima}%
    \endgroup
}
```

`\heightOf` `\widthOf` `\llxOf` `\llyOf` `\urxOf` `\uryOf` `\csOf`

More helper commands for calculating the height, width and path of an embedded file. These can be used by the user, that's you.

```
\def\heightOf#1{\csname#1heightOf\endcsname}
\def\widthOf#1{\csname#1widthOf\endcsname}
\def\bboxOf#1{\csname#1BBox\endcsname}
\def\llxOf#1{\csname#1Gin@llx\endcsname}
\def\llyOf#1{\csname#1Gin@lly\endcsname}
\def\urxOf#1{\csname#1Gin@urx\endcsname}
\def\uryOf#1{\csname#1Gin@ury\endcsname}
```

Use `\csOf` to expand a name.

```
\let\csOf\@nameuse
```

Other internal commands that save info.

```
%\def\grcxsp@pathOf#1{\csname#1path\endcsname}
%\def\grcxsp@importSF#1{\csname#1importScaleFactor\endcsname}
```

We redefine a command from graphics. When testing the `draft` option, we had some problems with an underscore `\_` in the value of the `name` key, so we sanitize this character.

```
%\def\spx@sanitize{\catcode‘\_=12\relax}
%\def\Gin@i{%
%  \@ifnextchar[%]
%    {\spx@sanitize\Gin@ii}
%    {\Gin@bboxfalse\Ginclude@graphics}}
```

`\embedEPS`

This is the command for embedding an EPS file in the document for use by the **SP** operator. The command takes three arguments, one of which is optional

`[#1]`: Recognizes two key-value pairs (1) `hiresbb`, this is the same key-value used by the graphicx package; (2) `transparencyGroup`, a new option for creating a transparency group. Using `transparencyGroup` without any value will make the embedded graphic into a transparency group, with a value adds additional keys as documented in the *PDF Reference*.

`#2`: A symbolic name for the embedded graphic, this name is used by distiller.

`#3`: path to the `EPS` file (without extension).

I prefer the `\embedEPS` commands to appear in the preamble, but they can appear anywhere before the first appearance of `\includegraphics` or `\insertEPS` that reference the embedded file. I suppose this embedding could have been automatic at the first occurrence of `\includegraphics` or `\insertEPS`, but I didn't go that route.

```
\newcommand{\embedEPS}[3][]{%
    \@ifundefined{#2Gin@llx}{}{%
        \PackageError{graphicxsp}%
        {The name, #2, on line \the\inputlineno\MessageBreak
        is already defined. All embedded graphics\MessageBreak
        must be assigned a unique name}
        {Give this embedded graphic a unique name.}%
    }%
    \begingroup
    \let\Gin@transparencygroup\@empty
```

We use the graphicx command `\Gread@eps` to verify that the graphic exists, and if so, get its bounding box parameters. We work only with `.eps` files so let's add the extension.

```
    \let\input@path\Ginput@path
    \filename@parse{#3.eps}%
    \Gin@getbase{.eps}%
    \@ifundefined{Gin@base}{%
        \PackageError{graphicxsp}%
        {%
            Graphics file #3 specified on \the\inputlineno\MessageBreak
             was not found%
        }{%
            Verify the file exists, is an eps file,\MessageBreak
            is on the latex search path, or is in the\MessageBreak
            current directory.%
        }%
    }{}%
    \e@namedef{gxsp@Gin@base}{\Gin@base}%
    \Gread@eps{\gxsp@Gin@base.eps}%
```

Now set the keys. We delayed the `\setkeys` because `name=#2` would set the switch `\if@Ginnamed` to `true`, which has consequences on computing the `\Gin@base` when `\Gin@setfile` is executed.

```
    \setkeys{Gin}{name=#2,#1}%
```

Once the file is found and the bounding box parameters are recorded by graphicx, we save these under the graphic's embedded symbolic name.

```
    \grcxsp@setPictureDimen%
        {\Gin@name}{\Gin@llx}{\Gin@lly}{\Gin@urx}{\Gin@ury}%
%   \x@namedef{\Gin@name path}{#3}%
```

If an embedded graphic exceeds the boundaries of the paper size, the graphic is clipped off. What I am doing below is determining the largest scale factor, `\gxsp@embedSF`, needed to embed the file without exceeding the page boundaries.

```
    \def\gxsp@embedSF{1}%
    \@tempdima=\Gin@urx bp
    \advance\@tempdima-\Gin@llx bp
    \ifdim\@tempdima>\paperwidth
        \Gscale@div\gxsp@embedSF\paperwidth\@tempdima
        \@tempdima=\Gin@ury bp
        \advance\@tempdima-\Gin@lly bp
        \@tempdima=\gxsp@embedSF\@tempdima
        \ifdim\@tempdima>\paperheight
          \edef\gxsp@embedSFSave{\gxsp@embedSF}%
          \Gscale@div\gxsp@embedSF\paperheight\@tempdima
          \@tempdima=\gxsp@embedSFSave\p@
          \@tempdima=\gxsp@embedSF\@tempdima
          \edef\gxsp@embedSF{\strip@pt\@tempdima}%
        \fi
    \else
        \@tempdima=\Gin@ury bp
        \advance\@tempdima\Gin@lly bp
        \ifdim\@tempdima>\paperheight
            \Gscale@div\gxsp@embedSF\paperheight\@tempdima
        \fi
    \fi
```

Now that we have `\gxsp@embedSF`, we add the current graphic to our collection of embedded files using `\AddToEmbeddedEPSs`, which is a variation on `\AddToShipoutPicture`, but uses `\ESO@AeBip@Hook` for our private use. We expand some of the arguments before executing `\AddToEmbeddedEPSs`.

```
    \edef\@tempa{%
    \noexpand\AddToEmbeddedEPSs{\noexpand\AtPageLowerLeft%
        {\noexpand\scalebox{\gxsp@embedSF}%
        {\noexpand\gxsp@embedEPS{\gxsp@Gin@base}{\Gin@name}}}%
        \noexpand\AtPageCenter{\noexpand\gcxsp@wrapEmbeddedFigure%
        {\Gin@transparencygroup}{#2}{\Gin@transparency}}}}\@tempa
    \endgroup
}
\@onlypreamble{\embedEPS}
```

In a dvi previewer, the embedded graphics are visible on the first page. The `\grcxsp@coverEmbeds` puts a white color box over the graphics, LaTeX content and other graphics are placed over this white color box. The white color box can be removed with the `showembeds` option.

```
\def\grcxsp@coverEmbeds{%
    \AddToEmbeddedEPSs{\AtPageLowerLeft{\colorbox{white}{%
    \parbox[b][\paperheight]{\paperwidth}{\hfill\vfill}}}}
\if\gxsp@showembeds1%
\AtBeginDocument{\grcxsp@coverEmbeds}
\else
\let\grcxsp@coverEmbeds\relax
\fi
```

`\gxsp@embedEPS`

The `\gxsp@embedEPS` command embeds the file, and is called by `\embedEPS`. It takes three options: (1) the value of `transparencyGroup`; (2) the EPS path; and (3) the symbolic name for the graphic.

The bounding box `/BBox` acts as a clipping path, if the graphic falls outside the box, it is clipped off. Since we don't know the size of the graphic in advance, and the value of the `%%BoundingBox` can be deceiving, set the of `/BBox` to an array with enormous dimensions, the default is `\grcxsp@maxDim = 5000`. This can be reset to larger value if you are embedding graphics of even more enormous dimensions.

```
\def\grcxsp@maxDim{5000}
```

Now, for the `\gxsp@embedEPS` command that embed the graphic between **BP** and **EP**.

```
\newcommand{\gxsp@embedEPS}[2]{%
    \gxsp@literalps@out{gsave \gxsp@setPSCoor
    \gxsp@psMrk/BBox [-\grcxsp@maxDim\space-\grcxsp@maxDim\space
      \grcxsp@maxDim\space\grcxsp@maxDim]\space/_objdef {Embedded:#2}
    /BP pdfmark grestore}%
    \message{<Embedding #1>}%
```

If we are using dvipsone, we can suppress the preview of the embedded file by not using the extension. Dviwindo will look for a tiff file, if not present, will not display a preview.

```
    \includegraphics{#1}%
    \gxsp@literalps@out{\gxsp@psMrk/EP pdfmark}%
}
```

We create a wrapper that shows the Embedded file under the original symbolic name Here we introduce any transparency ordered up in the option list of `\embedEPS`

```
\def\gcxsp@wrapEmbeddedFigure#1#2#3{%
    \def\Gin@transparencygroup{#1}\def\Gin@transparency{#3}%
    \gxsp@literalps@out{gsave \gxsp@setPSCoor
    \ifGin@clip
        [/BBox [\llxOf{#2}\space\llyOf{#2}\space
         \urxOf{#2}\space\uryOf{#2}]
    \else
        [ /BBox [-\grcxsp@maxDim\space-\grcxsp@maxDim\space
          \grcxsp@maxDim\space\grcxsp@maxDim]
    \fi\space /_objdef {#2}
    \ifx\Gin@transparencygroup\@empty\else
        \ifx\Gin@transparencygroup\Gin@exclamation
```

```
            /Group << /S/Transparency >>%
        \else
            /Group << /S/Transparency \Gin@transparencygroup >>%
        \fi
    \fi\space
    /BP pdfmark
        \gxsp@psMrk{Embedded:#2} /SP pdfmark
    \gxsp@psMrk/EP pdfmark
    grestore}%
}
```

`createImage` The `createImage` environment can be used for two purposes:

1. Use it to take a file already embedded, manipulate it, and give it a symbolic name.
2. Use postscript graphic operators to create an image.

The images can be shown using `\includegraphics` or `\insertEPS`, or they can be referenced as an appearance of a form field.

We try something different. My usual approach for a verbatim environment is to write the contents to an auxiliary file and input that file back in. This approach precludes using the environment in another command. The text to this environment should be PostScript or PDF language statements, or TEX macros that expand to same. We'll absorb the contents in the environment as an argument `#1` of the `\grxcsp@createImage` command. However, before we get to `\grxcsp@createImage` we must execute `\createImage`, the user's access to this code.

`\createImage` takes three arguments, the first one of which is optional

- `[#1]`: Takes the key-values of `\includegraphics`, plus some of the graphicxsp key-values, such as `transparencyGroup`. The name key is ignored, and is declared in the third parameter.
- `#2`: The bounding box for this image.
- `#3`: The `name` to be attached to this image.

```
\def\ci@undef@msg#1{\PackageWarning{graphicxsp}{The command
  `\expandafter\string\csname #1\endcsname'
  is already defined\MessageBreak
  choose a different name instead of\MessageBreak`#1'}}
\newcommand{\sp@createImage}[3][]{%
    \@ifundefined{#3}{}{\ci@undef@msg{#3}}%
    \x@namedef{#3}{#3}%
    \@ifundefined{#3Gin@llx}{}{%
        \PackageError{graphicxsp}%
        {The name, #3, on line \the\inputlineno\space\MessageBreak
        is already defined. All embedded graphics\MessageBreak
        must be assigned a unique name}
```

```
      {Give this embedded graphic a unique name.}%
   }%
  \setkeys{Gin}{#1}\def\Gin@name{#3}%
   \edef\@gtempa{#2 }%
  \expandafter\Gread@parse@bb\@gtempa \\%
  \begingroup\grxcsp@createImage
}
\let\postEP\@empty
\long\def\grxcsp@createImage#1\end#2{%
   \def\reserved@a{#2}\ifx\reserved@a\@currenvir
      \end{#2}\else\@badend{#2}\fi
   \edef\temp@transparencyGroup{%
   \ifx\Gin@transparencygroup\@empty\else
      \ifx\Gin@transparencygroup\Gin@exclamation
         /Group << /S/Transparency >>%
      \else
         /Group << /S/Transparency \Gin@transparencygroup >>%
      \fi
   \fi}%
   \grcxsp@setPictureDimen%
      {\Gin@name}{\Gin@llx}{\Gin@lly}{\Gin@urx}{\Gin@ury}%
```

(2009/02/19) We use `\AddToEmbeddedEPSs` to embed EPS created by the `createImage` environment. This allows the EPS to be used on the first page, which has been a problem in the past.

```
   \edef\@tempa{%
   \noexpand\AddToEmbeddedEPSs{\noexpand\AtPageLowerLeft{%
      \noexpand\gxsp@literalps@out{gsave \gxsp@setPSCoor
      \gxsp@psMrk/BBox
        [\Gin@llx\space\Gin@lly\space\Gin@urx\space\Gin@ury]
      /_objdef {\Gin@name} \temp@transparencyGroup\space/BP pdfmark
      \ifx\Gin@transparency\@empty\else
      \gxsp@psMrk\Gin@transparency\space/SetTransparency pdfmark\fi
      {#1} ?pdfmark
      \gxsp@psMrk/EP pdfmark
      grestore
      }%
   }}}\@tempa
   \endgroup
}
\let\createImage\sp@createImage
\let\endcreateImage\endsp@createImage
\@onlypreamble{\createImage}
```

`\insertEPS` The idea was to use the `\includegraphics` command to show a graphic that has been earlier embedded. However, one of `\includegraphics` arguments is the path of the eps file. Once, the file is embedded, the path is not needed, so this package defines `\insertEPS`. This command takes two arguments: (1) The usual `\includegraphics` options, plus any other options defined in this package; (2) the symbolic name. Because the symbolic name is passed as the second argument, it is not necessary to specify in the optional parameter list. The following two (should) be equivalent:

```
\embedEPS{myCoolSelfPic}
...
\begin{document}
...
\includegraphics[name=AdobeDon,width=1in]{myCoolSelfPic}
\insertEPS[width=1in]{AdobeDon}
...
```

```
\def\xsp@sanitize{\catcode`\_=12\relax}
\newcommand{\insertEPS}{\bgroup\xsp@sanitize
    \@ifstar
    {\Gin@cliptrue\let\gcxsp@star*\gcxsp@insertEPS}%
    {\Gin@clipfalse\let\gcxsp@star\@empty\gcxsp@insertEPS}}
\newcommand{\gcxsp@insertEPS}[2][]%
    {\expandafter\includegraphics\gcxsp@star[name=#2,#1]{}\egroup}
```

8 Messing with graphicx

In this section, we add some options to the graphicx package. We define some additional keys that will be recognized by `\includegraphics`. We also redefine `\Gin@ii` and `\Gin@setfile`, which are graphicx commands to make things work for us.

`name` Use the `name` key-value pair only for graphics already embedded by `\embedEPS`. When this key is present, we `\let \Ginclude@eps` to `\Ginclude@eps@SP`. `\Ginclude@eps` is the usual way of handling EPS files, `\Ginclude@eps@SP` is how we are to handle files already embedded. Usage:

```
\includegraphics[name=AdobeDon,width=1in]{myCoolSelfPic}
```

```
\newif\if@Ginnamed\@Ginnamedfalse
\define@key{Gin}{name}[]{\def\Gin@name{#1}%
    \@Ginnamedtrue\let\Ginclude@eps\Ginclude@eps@SP}
\def\Gin@name{}
```

`transparencyGroup` This defines the transparencyGroup key which is used only recognized with `\embedEPS`. See the Transparency section of the pdfmark Reference and the chapter on Transparency in the PDF Reference. In particular, see PDF Ref Table 7.13. Usage:

```
\embedEPS[transparencyGroup]{myCoolSelfPic}
```

```
\define@key{Gin}{transparencyGroup}[!]{\def\Gin@transparencygroup{#1}}
\def\Gin@transparencygroup{}
```

`transparency` Enter any transparency postscript key-value pairs for this image. These are ignored unless the embedded file is a transparency group, and you distill with `<< /AllowTransparency true >>` setdistillerparams!. Usage:

```
\embedEPS[transparencyGroup]{myCoolSelfPic}
...
\begin{document}
...
\includegraphics[name=AdobeDon,width=1in,
    transparency={/ca .5 /BM/Normal}]{myCoolSelfPic}
```

or

```
\insertEPS[width=1in,transparency={/ca .5 /BM/Normal}]{AdobeDon}
```

```
\define@key{Gin}{transparency}[]{\def\Gin@transparency{#1}}%
\def\Gin@transparency{}%
\define@key{Gin}{SMask}[]{\def\Gin@SMask{#1}}%
\def\Gin@SMask{}%
```

`presp`
`postsp`

We define two additional keys for creating special effects. The value of `presp` and `postsp` are postscript commands for manipulating the image. As the names suggest, `presp` is placed before the **SP** operator, and `postsp` is placed after. Example of usage is given in one of the demo files.

```
\define@key{Gin}{presp}{\def\Gin@presp{#1}}
\def\Gin@presp{}
\define@key{Gin}{postsp}{\def\Gin@postsp{#1}}
\def\Gin@postsp{}
```

The following key-value pairs are recognized by `\setSMask`, `\embedEPS` and `\includegraphics` and are used to set up a soft mask.

`SMask`
`subtype`
`group`
`bc`
`tc`

The key `SMask` is used in the optional parameter list of `\insertEPS` and `\includegraphics`, when that graphic is to use a soft mask. The value of `SMask` is a key-value list, the keys are `subtype`, `group`, `bc` and `tr`. The default for `subtype` is `Luminosity`, the other value recognized is `Alpha`. If `subtype` is not listed, `Luminosity` is used for the `subtype`. The `group` key is required, and the latex compile will stop if it is not specified. The value of group is the name of a graphic to be used as a mask. This graphic must be a transparency group with the `CS` key specified. The other two keys, `bc` (component color) and `tr` (transfer function) to complete the supported keys. See Table 7.10, page 553, of the *PDF Reference*, Version 8, for detailed descriptions of these key-values.

```
\define@key{Gin}{SMask}[]{\def\GinSP@SMask{#1}}
\def\GinSP@SMask{}
\define@key{GinSP}{subtype}[Luminosity]{\def\GinSP@subtype{#1}}
\def\GinSP@subtype{Luminosity}
\define@key{GinSP}{group}[]{\def\GinSP@group{#1}}
\def\GinSP@group{}
\define@key{GinSP}{bc}[]{\def\GinSP@bc{#1}}
\def\GinSP@bc{}
\define@key{GinSP}{tr}[]{\def\GinSP@tr{#1}}
\define@key{GinSP}{None}[None]{\def\SMaskSP@None{#1}}
\def\GinSP@tr{}
\def\SMaskSP@None{}
```

```
\def\SMaskSP@Identity{Identity}
```

`\setSMask` Use `\setSMask` to set a soft mask. This command takes one required argument, the name of the transparency group to be use as the source of alpha or color values for deriving the mask. The optional parameter consists of key-value pairs for the soft-mask dictionary, see table 7.10 of the PDF Reference, Version 8.

```
\def\sp@setSMask{%
  \ifx\GinSP@SMask\SMaskSP@None\gxsp@psMrk/SMask/None
    /SetTransparency pdfmark
  \else
    \gxsp@psMrk/SMask << /S/\GinSP@subtype\space
    \ifx\GinSP@bc\@empty\else/BC\GinSP@bc\space\fi
    \ifx\GinSP@tr\@empty\else\ifx\GinSP@tr\SMaskSP@Identity%
        /TR/Identity\else/TR {\GinSP@tr}\space\fi\fi
    /G {\GinSP@group} >> /SetTransparency pdfmark
  \fi
}
\newcommand{\setSMask}[2][]{%
        \setkeys{GinSP}{#1}\def\GinSP@group{#2}%
        \special{ps: \sp@setSMask}%
}
```

We redefine `\Gin@setfile`. If the graphic is named, we salt things with the bounding box parameters.

```
\def\Gin@setfile#1#2#3{%
  \ifx\\#2\\\Gread@false\fi
  \ifGin@bbox\else
    \ifGread@
\if@Ginnamed %dps (08/16/07)
    \edef\Gin@llx{\csname\Gin@name Gin@llx\endcsname}%
    \edef\Gin@lly{\csname\Gin@name Gin@lly\endcsname}%
    \edef\Gin@urx{\csname\Gin@name Gin@urx\endcsname}%
    \edef\Gin@ury{\csname\Gin@name Gin@ury\endcsname}%
\else
      \csname Gread@%
         \expandafter\ifx\csname Gread@#1\endcsname\relax
           eps%
         \else
           #1%
         \fi
      \endcsname{\Gin@base#2}%
\fi
    \else
      \Gin@nosize{#3}%
    \fi
  \fi
  \Gin@viewport@code
  \Gin@nat@height\Gin@ury bp%
  \advance\Gin@nat@height-\Gin@lly bp%
```

```
   \Gin@nat@width\Gin@urx bp%
   \advance\Gin@nat@width-\Gin@llx bp%
   \Gin@req@sizes
   \expandafter\ifx\csname Ginclude@#1\endcsname\relax
     \Gin@drafttrue
     \expandafter\ifx\csname Gread@#1\endcsname\relax
       \@latex@error{Can not include graphics of type: #1}\@ehc
       \global\expandafter\let\csname Gread@#1\endcsname\@empty
     \fi
   \fi
   \leavevmode
   \ifGin@draft
       \hb@xt@\Gin@req@width{%
         \vrule\hss
         \vbox to \Gin@req@height{%
            \hrule \@width \Gin@req@width
            \vss
            \if@Ginnamed %dps (08/18/07)
```

If the shownonames option is taken, we do not show the name of the graphic.

```
            \if@spx@shownameindraft
            \rlap{ \ttfamily\Gin@name}\fi
            \else
            \edef\@tempa{#3}%
            \rlap{ \ttfamily\expandafter\strip@prefix\meaning\@tempa}%
            \fi
            \vss
            \hrule}%
         \hss\vrule}%
   \else
\if@Ginnamed\else % dps (08/16/07)
     \@addtofilelist{#3}%
     \ProvidesFile{#3}[Graphic file (type #1)]%
\fi
     \setbox\z@\hbox{\csname Ginclude@#1\endcsname{#3}}%
     \dp\z@\z@
     \ht\z@\Gin@req@height
     \wd\z@\Gin@req@width
     \ifpreview{\setlength{\fboxsep}{0pt}\fbox{\box\z@}}\else\box\z@\fi%
   \fi}
\def\Gin@getbase#1{%
   \edef\Gin@tempa{%
     \def\noexpand\@tempa####1#1\space{%
       \def\noexpand\Gin@base{####1}}}%
```

If the current graphic is named, then we don't need to read the bounding box again or to see if it exists again.

```
     \if@Ginnamed
         \edef\Gin@ext{#1}\edef\Gin@base{\Gin@name}%dps (08/18/07)
     \else
```

If not named, we need to handle it in the usual way.

(2019/11/13) The latest LaTeX core defines `\IfFileExists@` which broke this package, we try a fix here.

```
        \IfFileExists{\filename@area\filename@base#1}%
        {\@ifundefined{IfFileExists@}{}%
         {\edef\@filef@und{\filename@area\filename@base#1 }}
         \Gin@tempa\expandafter\@tempa\@filef@und
         \edef\Gin@ext{#1}}{}%
    \fi
}
```

`\Gin@computeSF` Based on scaling info provided by graphicx, we compute the scale factors we need.

```
\def\Gin@computeSF{\def\@tempa{!}%
    \edef\gxsp@scaleFactor@x{\Gin@scalex}%
    \edef\gxsp@scaleFactor@y{\Gin@scaley}%
    \ifx\Gin@scaley\@tempa      % proportional height
        \ifx\Gin@scalex\@tempa  % proportional width
            \def\gxsp@scaleFactor@x{1}%
            \def\gxsp@scaleFactor@y{1}%
        \else                   % specified width
            \edef\gxsp@scaleFactor@y{\Gin@scalex}%
        \fi
    \else                       % specified height
        \ifx\Gin@scalex\@tempa  % proportional width
            \edef\gxsp@scaleFactor@x{\Gin@scaley}%
        \fi
    \fi
}
```

`\Ginclude@eps@SP` This is the substitute for the usual way of handing an EPS file. Here we use the **SP** to show the embedded graphic.

```
\def\gxsp@setBBox{%
    \ifGin@clip
        [/BBox [\Gin@llx\space\Gin@lly\space\Gin@urx\space\Gin@ury]
    \else
        [/BBox [-\grcxsp@maxDim\space-\grcxsp@maxDim\space
                \grcxsp@maxDim\space\grcxsp@maxDim]
    \fi
}
\def\Ginclude@eps@SP#1{%
% \message{<#1>}%
  \bgroup
```

See if the user has specified the `SMask` key, if yes, we'll check to see if a group name was specified. The group name is required. If no, we halt the compile job.

```
    \ifx\GinSP@SMask\@empty\else
        \edef\sp@expand@temp{\noexpand\setkeys{GinSP}{\GinSP@SMask}}%
        \sp@expand@temp
        \ifx\SMaskSP@None\@empty\ifx\GinSP@group\@empty
    \PackageError{graphicxsp}{The group key is required when you
    specify a SMask.}{Specify a group name for the group key.}%
\fi\fi\fi
\Gin@computeSF
\ifGin@bbox
\gxsp@literalps@out{%
    gsave \gxsp@setPSCoor
```

If `SMask` is specified, we call `\sp@setSMask` to set the graphics state parameters for a soft mask.

```
        \ifx\GinSP@SMask\@empty\else\sp@setSMask\fi
        \gxsp@setBBox\space
```

We push the basename (`\Gin@name:bbox`) and call `grcxspObjDef`. This procedure returns (`\Gin@name:bboxgraphicxspCnt-currentpage`). `graphicxspretn` then takes that result, and converts it to a name type. We then use it in `/_objdef {//graphicxspretn}`, using immediate execution.

```
        (\Gin@name:bbox@) grcxspObjDef
        /graphicxspretn exch cvx cvn def
        /_objdef {//graphicxspretn} /BP pdfmark
        \ifx\Gin@transparency\@empty\else
        \gxsp@psMrk\Gin@transparency\space
          /SetTransparency pdfmark\fi
        \gxsp@psMrk{\Gin@name} /SP pdfmark
        \gxsp@psMrk/EP pdfmark
        \gxsp@scaleFactor@x\space\gxsp@scaleFactor@y\space scale
        -\Gin@llx\space -\Gin@lly\space moveto
        currentpoint translate
        \Gin@presp
        \gxsp@psMrk{//graphicxspretn} /SP pdfmark
       \Gin@postsp
   grestore
}%
\else
\gxsp@literalps@out{%
    gsave \gxsp@setPSCoor
```

If `SMask` is specified, we call `\sp@setSMask` to set the graphics state parameters for a soft mask.

```
        \ifx\GinSP@SMask\@empty\else\sp@setSMask\fi
        \gxsp@setBBox\space
        (\Gin@name:grxsp@) grcxspObjDef
        /graphicxspretn exch cvx cvn def
        /_objdef {//graphicxspretn} /BP pdfmark
        \ifx\Gin@transparency\@empty\else
        \gxsp@psMrk\Gin@transparency\space
          /SetTransparency pdfmark\fi
        \gxsp@psMrk{\Gin@name} /SP pdfmark
        \gxsp@psMrk/EP pdfmark
        \gxsp@scaleFactor@x\space\gxsp@scaleFactor@y\space scale
        \ifx\Gin@viewport@code\relax\else
```

```
           -\Gin@llx\space-\Gin@lly\space moveto
            currentpoint translate\fi
          \Gin@presp
           \gxsp@psMrk{//graphicxspretn} /SP pdfmark
          \Gin@postsp
      grestore
  }%
  \fi
\egroup}
```

⟨/package⟩

9 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.

10 Change History

v.6 (2008/06/15)
- `createImage`: Made `createImage` into a private environment, `sp@createImage`, which can be used anywhere, then created a public version, `createImage`, which is restricted to the preamble. This change is needed to create dynamic appearances in the `rmannot` package. 11

v.7b (2012/09/11)
- `shownonames`: Added `draft`, `final`, `shownonames` options 2

v1.0.2 (2018/11/20)
- `\csOf`: Added `\csOf` 7
- `createImage`: Added convenience command to reference xobjects using commands 11

v1.0.3 (2019/11/13)
- General: latex core defines `\IfFileExists@` which broke this package, we try a fix here. 17

v1.0a (2017/03/12)
- `showembeds`: Added `\previewOn` and `\previewOff` 2