Tutorials

Using EPS Graphics in LaTeX 2ε Documents Part 1: The graphics and graphicx packages

Keith Reckdahl

Abstract

This is the first of two papers that explain how to use Encapsulated PostScript (EPS) files in LaTeX 2ε documents.

The graphics and graphicx packages provide commands which insert, scale, and rotate EPS graphics. In addition to graphic-insertion commands, commands which are commonly used in conjunction with EPS graphics are covered.

Compressed EPS files and non-EPS graphic formats (TIFF, GIF, JPEG, PICT, etc.) can also be inserted when `dvips` is used.

Since neither LaTeX nor `dvips` has any built-in decompression or graphics-conversion capabilities, that software must be provided by the user.

The second paper in the series, to appear later this year, will cover

- floating figures in various configurations (including more than one figure in a single float), and the use of the subfigure package,
- creation of boxed figures, by use of the `\fbox` command, or of the facilities of the fancybox package,
- manipulation of the caption of a figure, including use of the caption2 package, and
- modifying the text within an EPS file by using the PSfrag system, for example to include mathematical symbols or equations.

1 Introduction

Inserting Encapsulated PostScript (EPS) graphics in LaTeX originally required the low-level `\special` command. To make graphic-insertion easier and more portable, two higher-level packages epsf and psfig were written for LaTeX 2.09. In epsf, the graphics insertion was done by the `\epsfbox` command, while three other commands controlled graphic scaling. In psfig, the `\psfig` command not only inserted graphics, it also scaled and rotated them. While the `\psfig` syntax was popular, its code was not as robust as `\epsfbox`. The epsfig package was created as a hybrid of the two graphics packages, with its `\epsfig` command using the `\psfig` syntax and much of the more-robust `\epsfbox` code. Unfortunately, `\epsfig` still used some of the less-robust `\psfig` code.

The epsfig package was updated to LaTeX 2ε as a stop-gap measure while the LaTeX3 team addressed the general problem of inserting graphics in LaTeX 2ε. The resulting "graphics bundle" was totally re-written, and its commands are more efficient and more robust.

The graphics bundle contains the "standard" graphics package and the "extended" graphicx package. Both packages contain an `\includegraphics` command which includes graphics, but they contain *different* versions of `\includegraphics`. The syntax of the graphicx `\includegraphics` is modeled after `\psfig`, while the syntax of the graphics `\includegraphics` is modeled after the `\epsfbox` command. As a result, the `\includegraphics` command in graphicx supports scaling and rotating, but that in the graphics must be nested inside `\scalebox` and/or `\rotatebox` commands for scaling and/or rotating.

This paper has been typeset using the graphicx package because its syntax is more convenient than the graphics syntax. Since both packages have the same capabilities, the examples in this document can also be performed with the graphics package, although the resulting syntax may be more cumbersome. The syntax of the graphicx commands is described in section 5. The syntax of the graphics commands is described in section 6. For a full specification of the packages, see David Carlisle's graphics bundle documentation [1].

For backward compatibility, the graphics bundle also includes the epsfig package which replaces the original LaTeX 2ε epsfig package. The new epsfig package defines the `\epsfbox`, `\psfig`, and `\epsfig` commands as wrappers which translate to a simple call to the `\includegraphics` command.

2 LaTeX Terminology

A *box* is any LaTeX object (characters, graphics, etc.) that is treated as a unit (see [4, page 103]). Each box has a *reference point* on its left side, and a *baseline*, which is a horizontal line passing through the reference point (see Figure 1). When LaTeX forms lines of text, characters are placed left-to-right with their reference points aligned on a horizontal line called the *current baseline*, aligning the characters' baselines with the current baseline. LaTeX follows the same process for typesetting graphics or other objects; the reference point of each object is placed on the current baseline.

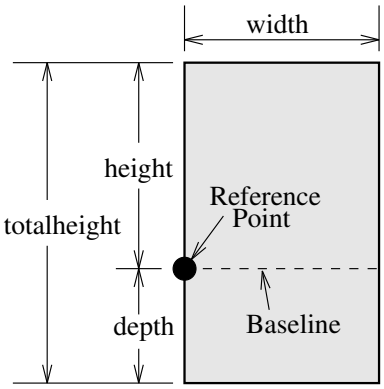


Figure 1: Sample LaTeX Box

The size of each box is described by the three lengths *height*, *depth*, *width*. The *height* is the distance from the reference point to the top of the box. The *depth* is the distance from the reference point to the bottom of the box. The *width* is the width of the box. The *totalheight* is defined as the distance from the bottom of the box to the top of the box, or totalheight = height + depth.

The reference point of a non-rotated EPS graphic is its lower-left corner (see left box in Figure 2), giving it zero depth and making its totalheight equal its height. The middle box in Figure 2 shows a rotated graphic where the height is not equal to the totalheight. The right box in Figure 2 shows a rotated graphic where the height is zero.

3 The EPS BoundingBox

In addition to PostScript graphics language commands which draw the graphics, EPS files contain a BoundingBox line which specifies the natural size of the graphics. By convention, the first line of a PostScript file specifies the type of PostScript and is then followed by a series of comments called the *header* or *preamble*. (Like LaTeX, PostScript's comment character is %). One of these comments specifies the BoundingBox. The BoundingBox line contains four numbers:

1. The x-coordinate of the lower-left corner of the BoundingBox.
2. The y-coordinate of the lower-left corner of the BoundingBox.
3. The x-coordinate of the upper-right corner of the BoundingBox.
4. The y-coordinate of the upper-right corner of the BoundingBox.

For example, here are the first 5 lines of an EPS file created by gnuplot:

```
%!PS-Adobe-2.0 EPSF-2.0
%%Creator: gnuplot
%%DocumentFonts: Times-Roman
%%BoundingBox: 50 50 410 302
%%EndComments
```

Thus the gnuplot EPS graphic has a lower-left corner with coordinates $(50, 50)$ and an upper-right corner with coordinates $(410, 302)$. The BoundingBox parameters have units of PostScript points which are $1/72$ of an inch, making the above graphic's natural width 5 inches and its natural height 3.5 inches.

Note that a PostScript point is is slightly larger than a TeX point which is $1/72.27$ of an inch. In TeX and LaTeX, PostScript points are called "big points" and abbreviated `bp` while TeX points are called "points" and abbreviated `pt`.

3.1 Converting PS files to EPS

While most PostScript files (without BoundingBox information) can be converted to EPS, there are restrictions on the PostScript commands which can be used in EPS files. For example, EPS files cannot include the `setpagedevice`, `letter`, or `a4` PostScript operators. Single-page PostScript files without any such offending commands can be converted to EPS by one of the following methods:

1. The best option is to use a utility such as ghostscript's `ps2epsi` which will read the PostScript file, calculate the BoundingBox parameters, and create an EPS file (complete with a BoundingBox) which contains the PostScript graphics. Unfortunately, ghostscript is a large package which is not trivial to install.

2. Alternatively, the BoundingBox parameters can be calculated and then either entered in the `bb` option of `\includegraphics` or a text editor can be used to insert them directly in the PostScript file's BoundingBox line. There are several ways to calculate the BoundingBox:

   (a) The `bbfig` script uses a PostScript printer to calculate the BoundingBox. `bbfig` adds some PostScript commands to the beginning of the PostScript file and sends it to the printer. At the printer, the added PostScript commands calculate the BoundingBox of the original PostScript file, printing the BoundingBox coordinates superimposed on the PostScript graphic.

   (b) Use ghostview to display the PostScript graphic. As you move the pointer around

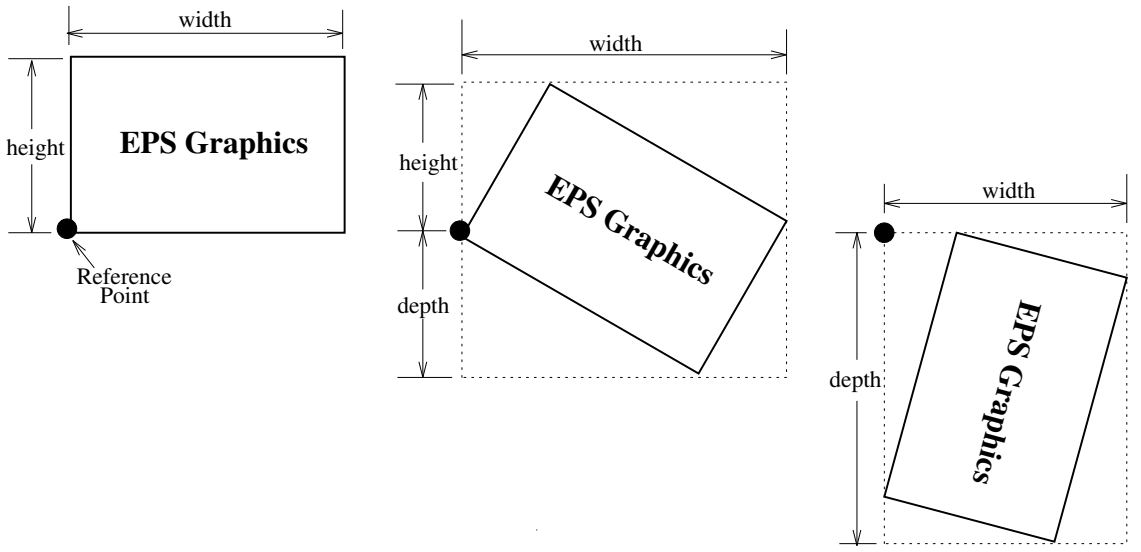


Figure 2: Rotated LaTeX Boxes

the graphic, ghostview displays its coordinates (with respect the the lower-left corner of the page). To determine the BoundingBox parameters, record the pointer coordinates at the lower-left corner of the graphic and the upper-right corner of the graphic.

(c) Print out a copy of the PostScript graphics and measure the horizontal and vertical distances (in inches) from the lower-left corner of the paper to the lower-left corner of the graphics. Multiply these measurements by 72 to get the coordinates of the BoundingBox's lower-left corner. Likewise, measure the distances from the lower-left corner of the paper to the upper-right corner of the graphics to get the coordinates of the BoundingBox's upper-right corner.

4 Graphics in DVI Files

When LaTeX documents are compiled, the graphics-inclusion commands do not insert the EPS graphics file into the DVI file. Rather, they do two things:

1. They reserve the proper amount of space for the graphic in the LaTeX document.
2. They place a file-specification command in the DVI file which specifies the name of the EPS file.

When a DVI-to-PS converter (such as `dvips`) converts the DVI file to PostScript, the file-specification command causes the converter to insert the EPS graphics into the PostScript file. Therefore,

- the EPS graphics do not appear in most DVI-viewers. To help the user with placement of the graphics, most DVI viewers display the BoundingBox in which the graphics will be inserted.
- the EPS files must be present when the DVI file is converted to PS. Thus the EPS files must accompany DVI files whenever they are moved.

5 The Commands in the graphicx Package

The coverage of the graphicx package is sporadic: [3] covers both the graphics and graphicx packages, [4] only covers the graphics package and [2] describes neither. The best reference for the graphics and graphicx packages is [1].

The graphicx package has five main commands:

```
\includegraphics[options]{filename}
\rotatebox{angle}{argument}
\scalebox{h-scale}[v-scale]{argument}
\resizebox{width}{height}{argument}
\resizebox*{width}{totalheight}{argument}
```

5.1 The includegraphics Command

Syntax:

```
\includegraphics[options]{filename}
```

Table 1 lists the geometric options available when using the `\includegraphics` command. The BoundingBox can alternatively be specified with the options `natheight` and `natwidth`. `natheight=h` with `natwidth=w` are equivalent to `bb=0 0 h w`. For backward compatibility, the BoundingBox coordinates can also be individually specified with `bbllx`, `bblly`, `bburx`, `bbury` options.

Table 1: `\includegraphics` Options

`height`	The height of the graphics (in any of the accepted TEX units).
`totalheight`	The totalheight of the graphics (in any of the accepted TEX units). *(Added 6/95)*
`width`	The width of the graphics (in any of the accepted TEX units).
`scale`	Scale factor for the graphic. Specifying `scale=2` makes the graphic twice as large as its natural size.
`angle`	Specifies the angle of rotation (in degrees) with a counter-clockwise (anti-clockwise) rotation being positive.
`origin`	The `origin` command specifies what point to use for the rotation origin. (By default, the object is rotated about its reference point.) *(Added 12/95)* The possible origin points are the same as those for the `\rotatebox` command in section 5.4. For example, `origin=c` rotates the graphic about its center.
`bb`	Specifies BoundingBox parameters. For example `bb=10 20 100 200` specifies that the BoundingBox has its lower-left corner at (10,20) and its upper-right corner at (100,200).

Table 2: `\includegraphics` Cropping Options

`viewport`	Specify what portion of the graphic to view. Like a BoundingBox, the area is specified by four numbers which are the coordinates of the lower-left corner and upper-right corner. The coordinates are relative to lower-left corner of the BoundingBox. *(Added 6/95)* For example, `viewport=0 0 72 72` displays the 1-inch square from the lower left of the graphic. Note that some early graphicx versions may have a broken `viewport` option in which `viewport=a b c d` produces an upper-right corner of (a+c,b+d) instead of (c,d).
`trim`	An alternate method for specifying what portion of the graphic to view. The four numbers specify the amount to remove from the left, bottom, right, and top side, respectively. Positive numbers trim from a side, negative numbers add to a side. *(Added 6/95)* For example, `trim=1 2 3 4` trims the graphic by 1 `bp` on the left, 2 `bp` on the bottom, 3 `bp` on the right, 4 `bp` on the top.

Table 3: `\includegraphics` Boolean Options

`clip`	When `clip=true` or `clip` is specified, any graphics outside of the viewing area are clipped and do not appear.
`keepaspectratio`	When `keepaspectratio=true` or `keepaspectratio` is specified, specifying both the `width` and `height` or `totalheight` options does *not* distort the graphic. Instead, the graphic is made as large as possible such that its aspect ratio remains the same and the graphic does not exceed either the specified height or width. *(Added 9/95)*
`draft`	When `draft=true` or `draft` is specified, the graphic's BoundingBox and filename is displayed in place of the graphic, saving time. See section 7.

Since `\includegraphics` automatically reads the BoundingBox parameters from the EPS file, these options are usually not specified. They are useful if the BoundingBox parameters in the EPS file are missing or are incorrect. While the `bb` option can also be used for cropping the EPS graphics, the `viewport` or `trim` options (see Table 2) are recommended. Table 3 lists other control options.

Example:

The EPS file `box.eps` contains:

The commands:

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
   Some text.
   \includegraphics{box.eps}
   More text.
\end{document}
```

produce:

Some text.  More text.

Since `\includegraphics` does not end the current paragraph, it can place EPS graphics within text, for example ⊗ or ◔. The placement of the graphic is controlled by the current text justification. To center the graphic, put it inside a center environment:

```
\begin{center}
   \includegraphics[width=2in]{box.eps}
\end{center}
```

Alternately, if the `\includegraphics` command is inside an environment (such as `minipage` or `figure`), the `\centering` declaration centers the remaining output of the environment. For example:

```
\begin{figure}
   \centering
   \includegraphics[width=2in]{box.eps}
\end{figure}
```

is similar to:

```
\begin{figure}
\begin{center}
   \includegraphics[width=2in]{box.eps}
\end{center}
\end{figure}
```

The difference between these examples is that the `center` environment produces extra vertical space above and below the environment, while `\centering` produces no extra space.

5.2 The scalebox Command

Syntax:

```
\scalebox{h-scale}[v-scale]{argument}
```

The `\scalebox` command scales an object, making its width be its original width multiplied by `h-scale`. The object can be any LaTeX object: letter, paragraph, EPS graphic, etc. The object's height is its original height multiplied by `v-scale`. Negative values reflect the object. If `v-scale` is omitted, it defaults to h-scale, which keeps the aspect ratio constant.

5.3 The resizebox Commands

Syntax:

```
\resizebox{width}{height}{argument}
\resizebox*{width}{totalheight}{argument}
```

The `\resizebox` command resizes an object to a specified size. The object can be any LaTeX object: letter, paragraph, EPS graphic, etc. Specifying `!` as either height or width makes that length be such that the aspect ratio remains constant. The standard LaTeX 2_ε arguments `\height`, `\depth`, `\width`, `\totalheight` can be used to refer to the original size of `argument`. So `\resizebox{2in}{\height}% {argument}` makes `argument` keep its same height but have a width of 2 inches.

The `\resizebox*` command only differs from `\resizebox` in its second argument, which specifies the `totalheight` of the object.

5.4 The rotatebox Command

Syntax:

```
\rotatebox[options]{angle}{argument}
```

The `\rotatebox` command rotates an object by an angle given in degrees, with a counter-clockwise rotation being positive. The object can be any LaTeX object: letter, paragraph, EPS graphic, etc. By default, the object is rotated about its reference point. The options allow the user to specify the point of rotation:

1. Specifying the `[x=xdim,y=ydim]`, the object is rotated about the point whose coordinates relative to the reference point are (`xdim,ydim`).
2. The `origin` option specifies one of 12 special points shown in in Figure 3.

   The horizontal position of the `origin` points is specified by one of three letters: `lcr` (which stand for left, center, right), while the vertical position is specified by one of four letters:

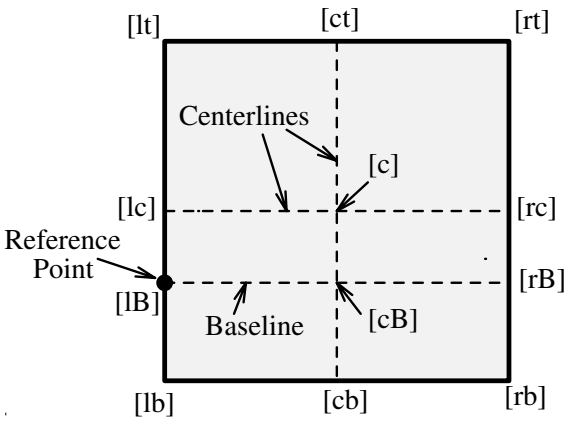


Figure 3: Available Origin Points

`t,c,B,b` (which stand for top, center, Baseline, bottom). For example:

`[rb]` specifies the bottom-right corner

`[lt]` specifies the top-left corner

`[cB]` specifies the center of the graphic's Baseline

`[lc]` specifies the midpoint of the left side

`[ct]` specifies the midpoint of the top side

Note that:

- The order of the letters is not important, making `[br]` equivalent to `[rb]`.
- `c` represents either the horizontal center or vertical center depending what letter is used with it.
- If only one letter is specified, the other is assumed to be `c`, making `[c]` equivalent to `[cc]`, `[l]` equivalent to `[lc]`, `[t]` equivalent to `[tc]`, etc.

6 The graphics Version of includegraphics

The graphics package contains two commands `\includegraphics` and `\includegraphics*` which are identical except that `\includegraphics*` clips (does not show) graphics outside the BoundingBox. The syntax for `\includegraphics` is:

```
\includegraphics[llx,lly][urx,ury]{filename}
```

`[llx,lly]` are the x and y coordinates of the lower-left corner of the image. `[urx,ury]` are the x and y coordinates of the upper-right corner of the image. If no coordinates are given, the BoundingBox in the file is used. If only one set of coordinates is given, it is assumed to be `[urx,ury]`, with `[llx,lly]` set to zero. The default units for the coordinates are `bp`, although any valid TEX units can be used.

The graphics package's `\rotatebox`, `\scalebox`, `\resizebox` commands are the same as the corresponding graphicx commands except that the graphics version of `\rotatebox` does not allow any of the options which the graphicx version offers (see section 5.4).

The following commands use the graphicx version of `\includegraphics`:

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
%% include file1.eps with a width of 3 inches
\includegraphics[width=3in]{file1.eps}

%% include file2.eps with a width of 3 inches,
%%    then rotate 45 degrees
\includegraphics[width=3in,angle=45]{file2.eps}

%% include file3.eps, rotate 45 degrees,
%%    and resize to a width of 3 inches
\includegraphics[angle=45,width=3in]{file3.eps}
\end{document}
```

The following commands use the graphics version of `\includegraphics` to produce the same output:

```
\documentclass{article}
\usepackage{graphics}
\begin{document}
%% include file1.eps with a width of 3 inches
\resizebox{3in}{!}{\includegraphics{file1.eps}}

%% include file2.eps with a width of 3 inches,
%%    then rotate 45 degrees
\rotatebox{45}{\resizebox{3in}{!}%
                {\includegraphics{file2.eps}}}

%% include file3.eps, rotate 45 degrees,
%%    and resize to a width of 3 inches
\resizebox{3in}{!}{\rotatebox{45}%
                {\includegraphics{file3.eps}}}
\end{document}
```

7 Draft Mode in graphicx

Since LaTeX documents containing PostScript figures take longer to display and print, it often is desirable to omit the actual graphic when preliminary versions of the document are viewed or printed. If the graphics or graphicx packages are used with the `draft` option:

```
\usepackage[draft]{graphicx}
```

then only the BoundingBox and name of any subsequent EPS graphics are displayed:

The graphicx version of `\includegraphics` has a `draft` option which allows the user to also control this feature for individual graphics. For example:

```
\includegraphics[draft,width=1.2in]{box.eps}
```

8 Specifying Height and/or Width in graphicx

The graphic's height and/or width can be specified, resulting in the following combinations:

- If neither the height nor the width is specified, the EPS graphic is included with its natural size (the size specified by the BoundingBox).
- If the height is specified and the width is not specified, the EPS graphic is included with the specified height and a width such that its height/width aspect ratio remains the same.
- If the width is specified and the height is not specified, the EPS graphic is included with the specified width and a height such that its height/width aspect ratio remains the same.
- If both height and width are specified:
  - If the `keepaspectratio` option is *not* specified, the EPS graphic is scaled anamorphically to fit both the specified height and width.
  - If the `keepaspectratio` option is specified, the graphic is made as large as possible such that its aspect ratio remains the same and the graphic does not exceed either the specified height or width.

The following LaTeX command makes the included graphic as wide as the text:

```
\includegraphics[width=\textwidth]{box.eps}
```

The following LaTeX command makes the included graphic 80% as wide as the text:

```
\includegraphics[width=0.80\textwidth]{box.eps}
```

The following commands make the width of the included graphic 2 inches less than the width of text:

```
\newlength{\epswidth}
\setlength{\epswidth}{\textwidth}
\addtolength{\epswidth}{-2.0in}
\includegraphics[width=\epswidth]{box.eps}
```

If the calc package is available, this is shortened to:

```
\newlength{\epswidth}
\setlength{\epswidth}{\textwidth -2.0in}
\includegraphics[width=\epswidth]{box.eps}
```

The `\newlength` command only needs to be issued once. Subsequent graphics can be scaled without reissuing the `\newlength` command. The length name `\epswidth` is not special. Any other name (which isn't already used by LaTeX) could have been used. The calc package with the 12/95 graphicx package shortens this further to:

```
\includegraphics[width=\textwidth-2.0in]%
                                        {box.eps}
```

8.1 Problems with Specifying Height

Users must be careful when using the `height` option. When users want to specify an object's "height", they often mean the overall height which is set by the `totalheight` option and not the `height` option. If the `height` option is mistakenly used instead of `totalheight`, the results may or may not be bad:

- If the object happens to have a zero depth (see the left box in Figure 2) the `totalheight` is the same as the `height` and everything works fine.
- If the object has non-zero depth (see the middle box in Figure 2) the object is scaled such that the object's `height` is as large as the desired `totalheight`, making the object too large.
- If the object has zero height (see the right box in Figure 2) an "Arithmetic overflow" (divide-by-zero) error occurs. This happens because LaTeX calculates the scaling factor as

$$\text{scaling} = \frac{\text{requested height}}{\text{height}}$$

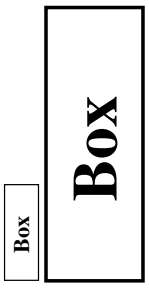
Note that the LaTeX2.09 `\psfig` command and early versions of `\includegraphics` only have a `height` option.

9 Rotation with Scaling in graphicx

Since the `\includegraphics` options are interpreted from left to right, the order in which the angle and size are specified makes a difference. For example:

```
\begin{center}
  \includegraphics[angle=90,%
          totalheight=0.5in]{box.eps}
  \includegraphics[totalheight=0.5in,%
                    angle=90]{box.eps}
\end{center}
```

produces:



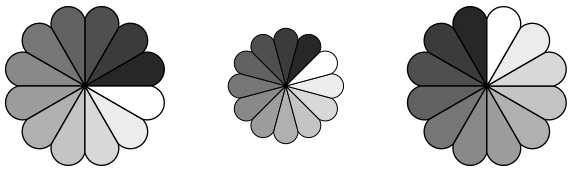
The first box is rotated 90 degrees and then scaled such that its height is a half inch. The second box is scaled such that its height is a half inch and then it is rotated 90 degrees.

9.1 Scaling of Rotated Graphics

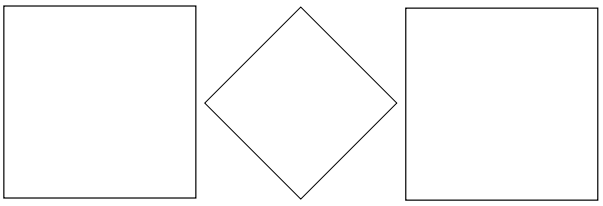
When the height or width of a graphic is specified, the specified size is not the size of the graphic but rather of its BoundingBox. This distinction is especially important in order to understand the scaling of rotated graphics. For example:

```
\begin{center}
  \includegraphics[totalheight=1in]%
                            {rosette.eps}
  \includegraphics[angle=45,totalheight=1in]%
                            {rosette.eps}
  \includegraphics[angle=90,totalheight=1in]%
                            {rosette.eps}
\end{center}
```

produces:



Although it may seem strange that the graphics have different sizes, it makes sense after viewing the BoundingBoxes:



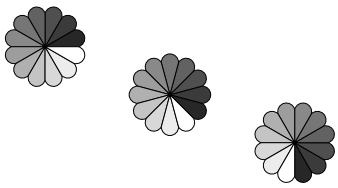
Each graphic is scaled such that its rotated BoundingBox is 1 inch tall.

9.2 Alignment of Rotated Graphics

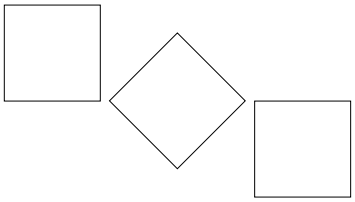
When graphics are rotated, the objects may not align properly. For example:

```
\begin{center}
  \includegraphics[totalheight=0.5in]%
                             {rosette.eps}
  \includegraphics[totalheight=0.5in,%
                   angle=-45]{rosette.eps}
  \includegraphics[totalheight=0.5in,%
                   angle=-90]{rosette.eps}
\end{center}
```

produces:



Again, this is better illustrated by the BoundingBoxes:



In this case, the objects' reference points (original lower-left corners) are aligned on a horizontal line. If it is desired to instead have the centers aligned, the `minipage` environment can be used:

```
\begin{center}
  \begin{minipage}[c]{0.625in}
    \centering
    \includegraphics[totalheight=0.5in,%
                     angle=0]{rosette.eps}
  \end{minipage}
  \begin{minipage}[c]{0.625in}
    \centering
    \includegraphics[totalheight=0.5in,%
                    angle=-45]{rosette.eps}
  \end{minipage}
  \begin{minipage}[c]{0.625in}
    \centering
    \includegraphics[totalheight=0.5in,%
                    angle=-90]{rosette.eps}
  \end{minipage}
\end{center}
```

However, an easier solution uses the `\rotatebox` command to rotate the graphic about its center:

```
\begin{center}
  \includegraphics[totalheight=0.5in]%
                              {rosette.eps}
  \rotatebox[origin=c]{-45}{%
     \includegraphics[totalheight=0.5in]%
                              {rosette.eps}}
  \rotatebox[origin=c]{-90}{%
     \includegraphics[totalheight=0.5in]%
                              {rosette.eps}}
\end{center}
```

This aligns the centers of the graphics:

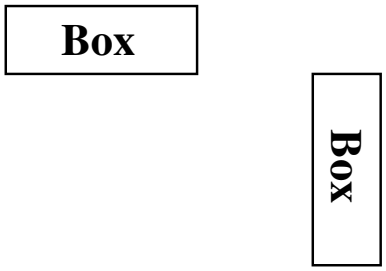
If the 12/95 version of graphicx is used, the `origin` option can be used in `\includegraphics`:

```
\begin{center}
  \includegraphics[totalheight=0.5in]%
                                {rosette.eps}
  \includegraphics[totalheight=0.5in,%
            origin=c,angle=-45]{rosette.eps}
  \includegraphics[totalheight=0.5in,%
            origin=c,angle=-90]{rosette.eps}
\end{center}
```

Similarly, the commands:

```
\begin{center}
    \includegraphics[width=1in]{box.eps}
    \hspace{0.5in}
    \includegraphics[width=1in,angle=-90]%
                                {box.eps}
\end{center}
```

produce:



while the following commands:

```
\begin{center}
  \includegraphics[width=1in]{box.eps}
  \hspace{0.5in}
  \rotatebox[origin=br]{-90}{%
    \includegraphics[width=1in]{box.eps}}
\end{center}
```

align the bottoms of the graphics:

If the 12/95 version of graphicx is used, the `origin` option can be used in `\includegraphics`:

```
\begin{center}
  \includegraphics[width=1in]{box.eps}
  \hspace{0.5in}
  \includegraphics[width=1in,origin=br,%
                   angle=-90]{box.eps}}
\end{center}
```

10 Compressed and Non-EPS Graphics Files

The commands `\DeclareGraphicsRule` and `\DeclareGraphicsExtensions` control how LaTeX deals with the files specified in `\includegraphics` commands.

When using `dvips`, users can specify an operation to be performed on the file before it is inserted. By making this operation a decompression command, compressed graphics files can be used. By making this a graphics-conversion command, non-EPS graphics files can be used. **Since `dvips` is currently the only DVI-to-PS converter with this capability, everything in this section requires `dvips`.**

`\DeclareGraphicsRule` has a 'command' argument; this is only usable on an operating system that supports pipes. Without piping, the decompression or conversion cannot be done on-the-fly and the user must store all graphics as uncompressed EPS files.

Depending on the system defaults, users may need to pass the `dvips` option to the graphicx package. This can be done by either specifying the `dvips` global option in the `\documentclass` command:

```
\documentclass[dvips,11pt]{article}
```

or by specifying the `dvips` option when loading the package using the `\usepackage` command:

```
\usepackage[dvips]{graphicx}
```

Since specifying the `dvips` as a global option passes it to all packages, it is generally preferred.

10.1 The DeclareGraphicsRule Command

The `\DeclareGraphicsRule` command specifies how `\includegraphics` should treat files depending on their extensions. Multiple `\DeclareGraphicsRule` commands may be issued. The syntax is:

```
\DeclareGraphicsRule{ext}{type}%
                    {sizefile}{command}
```

See Table 4 for details of the arguments to the command. For example, the following command:

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}%
                          {`gunzip -c #1}
```

specifies that any file with a `.eps.gz` extension is treated as compressed `eps` file. It also specifies that the BoundingBox information is stored in the file with a `.eps.bb` extension, and that `gunzip -c` command uncompresses the file. (Since LaTeX cannot read BoundingBox information from a compressed file, the BoundingBox line must be stored in an uncompressed file.)

Since the following graphics rules are defined by default in `dvips.def`, users generally do not need to use the `\DeclareGraphicsRule` command:

```
\DeclareGraphicsRule{.eps}{eps}{.eps}{}
\DeclareGraphicsRule{.ps}{eps}{.ps}{}
\DeclareGraphicsRule{.pz}{eps}{.bb}%
                          {`gunzip -c #1}
\DeclareGraphicsRule{.eps.Z}{eps}{.eps.bb}%
                          {`gunzip -c #1}
\DeclareGraphicsRule{.ps.Z}{eps}{.ps.bb}%
                          {`gunzip -c #1}
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}%
                          {`gunzip -c #1}
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}%
                          {`gunzip -c #1}
\DeclareGraphicsRule{.pcx}{bmp}{}{}
\DeclareGraphicsRule{.bmp}{bmp}{}{}
\DeclareGraphicsRule{.msp}{bmp}{}{}
\DeclareGraphicsRule{*}{eps}{*}{}
```

Table 4: `\DeclareGraphicsRule` Arguments

`ext`	The file extension.
`type`	The graphics type for that extension.
`sizefile`	The extension of the file which contains the BoundingBox information for the graphics. If this option is blank `{}`, the size information must be specified by an `\includegraphics` option.
`command`	The command to be applied to the file (often left blank `{}`). The command must be preceded by a single backward quote (not to be confused with the more common forward single quote).

The first two commands define the `.eps` and `.ps` extensions as `eps` files. The next five commands define extensions for compressed `eps` files. The next three commands define extensions for bitmaps (see section 10.3.1). The last command defines any other suffix as an `eps` file.

For example, to compress the EPS file `file.eps`, the BoundingBox line must first be extracted and stored in `file.eps.bb`. The EPS file can then be compressed by the `gzip file.eps` command (maximise compression by using `gzip -9 file.eps`). The old epsfig package came with `epsbb`, a `perl` script which creates the BoundingBox file and then compresses the EPS file. `epsbb` is still available from CTAN.

10.2 The DeclareGraphicsExtensions Command

The `\DeclareGraphicsExtensions` command tells LaTeX which extensions to try if a user specifies a file with no extension in the `\includegraphics` command. The following graphic extensions are defined by default in `dvips.def`:

```
\DeclareGraphicsExtensions{.eps,.ps,%
                .eps.gz,.ps.gz,.eps.Z}
```

With the above graphics extensions specified, the command `\includegraphics{file}` makes LaTeX first look for `file.eps`, then for `file.ps`, then for `file.eps.gz`, etc., until a file is found.

The `\DeclareGraphicsExtensions` command allows the graphics to be specified with

```
\includegraphics{file}
```

instead of

```
\includegraphics{file.eps}
```

The first syntax has the advantage that if you later decide to compress `file.eps`, you do not need to edit the LaTeX file.

10.3 Including Non-EPS Graphic Files

While it is easy to insert EPS graphics into LaTeX documents, it is not as straightforward to insert non-EPS graphics (GIF, TIFF, JPEG, PICT, etc.). A simple solution is to find out if the application which generated the non-EPS graphic also generates EPS output. If not, a graphics conversion program (such as ImageMagick, xv, netpbm, pbmplus) must be used to convert the graphics to PostScript.

Since a non-EPS graphics file may be smaller than the corresponding EPS file, it may be desirable to keep the graphics in a non-EPS format and convert them to PostScript when the DVI file is converted to PostScript. If `dvips` is used, this on-the-fly conversion can be specified by the command option in `\DeclareGraphicsRule`. For example, to use on-the-fly conversion to insert `file.gif` into a LaTeX document, one needs to:

1. Find a GIF-to-PS conversion program (assume it's called `gif2ps`)
2. One needs to create a `.bb` file which specifies the natural size of the `file.gif` graphics. To do this, convert `file.gif` to PostScript and:
   (a) If the Postscript file is EPS, save the BoundingBox line in `file.bb`
   (b) If the Postscript file is not EPS, determine the appropriate BoundingBox (see section 3) and store it in `file.bb`
3. Keep `file.gif` and delete the PostScript file.
4. Enter `\DeclareGraphicsRule{.gif}{eps}%` `{.bb}{‘gif2eps #1}` in the LaTeX document.

When the command `\includegraphics{file.gif}` is issued, LaTeX will read the BoundingBox from `file.bb` and will also tell `dvips` to use `gif2eps` to convert `file.gif` into PostScript.

10.3.1 Direct Support for Non-EPS Graphics

It is often requested that LaTeX and `dvips` support the direct inclusion of non-EPS graphic formats, making it as easy as inserting EPS files. While this would be convenient, there unfortunately are some problems which complicate things.

For example, most non-EPS graphic formats use binary files which cannot be read by TEX, which prevents LATEX from determining the size of the non-EPS graphics. Furthermore, supporting non-EPS graphics would also require `dvips` to incorporate graphics conversion capabilities (GIF-to-PS, TIFF-to-PS, etc.). This would not only require a lot of programming, it would also require more maintenance in the future.

Rather than directly incorporating graphics conversion routines, `dvips` provides a mechanism of calling external conversion programs. This mechanism can be accessed from LATEX by use the the command argument of `\DeclareGraphicsRule`. This has the benefit of being more flexible than direct support, and since it keeps the graphics conversion uncoupled from the DVI-to-PS conversion, users are free to choose their own graphics conversion program.

While LATEX and `dvips` generally do not support the direct inclusion of non-EPS graphics, there are some exceptions:

1. If `dvips` is compiled with `-Demtex`, it supports some EmTEX `\special` commands, allowing it to include PCX, BMP, or MSP bitmaps.
2. Some commercial versions of LATEX support non-EPS graphics:
   (a) *Textures* for the Macintosh supports PICT graphics.
   (b) Y&Y's TEX package for Windows includes the DVI-to-PS converter `DVIPSONE` which supports TIFF files. However, TEX cannot read the binary TIFF files, preventing LATEX from reading the TIFF tags the same way it reads EPS BoundingBox information. Since LATEX cannot determine the natural size of TIFF graphics, the user must still use a `.bb` file or specify the `bb` parameters explicitly in the optional argument of the `\includegraphics` command.

   Check your documentation or contact the company's customer service for the correct syntax.

References

[1] David P. Carlisle. Packages in the 'graphics' bundle. Available from CTAN as `grfguide.tex` or `grfguide.ps`.

[2] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LATEX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[3] Helmut Kopka and Patrick Daly. *A Guide to LATEX 2ε*. Addison-Wesley, Reading, Massachusetts, 1995.

[4] Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

◇ Keith Reckdahl
Stanford University
Box 9030
Palo Alto, CA 94309
USA
`reckdahl@leland.stanford.edu`