The combinedgraphics package*

Christian Schneider
<software(at)chschneider(dot)eu>

July 13, 2012

Contents

*This document corresponds to combinedgraphics v0.2.2, dated 2012/07/13.

Abstract

This package provides a macro (`\includecombinedgraphics`) for the inclusion of combined EPS/LaTeX and PDF/LaTeX graphics (an export format of Gnuplot, Xfig, and maybe other programs). Instead of including the graphics with a simple `\input`, the `\includecombinedgraphics` macro has some benefits:

- changing the font and color of the text of the LaTeX parts
- rescaling the graphics without affecting the font of the LaTeX parts
- automatic inclusion of the vector graphics parts, as far as LaTeX parts do not do it (e. g., for files exported from Gnuplot before version 4.2 or Xfig)
- changing the inclusion order of LaTeX and vector graphics parts (e. g., Gnuplot 4.4 exports files in a way that the vector graphics part overlays the LaTeX part, which means that text may be hidden under shaded areas)
- rescaling and rotating of complete graphics (similar to `\includegraphics` from graphicx package)
- scaling the vector graphics part (without affecting the font) to a defined width or height of the graphics, which might, e.g., be of importance for graphics that are intended for publication in a journal requesting a certain size of graphics

A test PDF file with extensively commented examples is shipped with this package and a BASH script (`texpic2epspdf.sh`) to convert combined EPS/LaTeX and PDF/LaTeX graphics to EPS or PDF files with text part included.

1 The Combined EPS/LaTeX format

Graphics in combined EPS/LaTeX or PDF/LaTeX format consist of two files:

1. an EPS or PDF file containing the vector graphics part
2. an LaTeX file containing the text part

This format has the advantage that it is possible to generate a high-quality vector graphics with text containing all symbols, macros and other stuff provided by LaTeX. The text is formatted by LaTeX itsself when including the graphics (*not* during creation of the graphics!) and thus it is possible to use everything provided by LaTeX up to the font of the document for text inside the graphics.

Typically, the LaTeX part of the graphics is included by `\input`. The vector graphics part is either included automatically by the LaTeX part with an `\includegraphics` command (or something similar) and overlayed by the text or the user will have to combine both parts manually like this (e. g., for gnuplot before version 4.2 or Xfig) for a pair of files `foobar.tex`/`foobar.eps`:

```
\begin{picture}(0,0)%
  \includegraphics{foobar}%
\end{picture}%
\input{foobar}
```

Although the format has some nice advantages compared to other formats, the user has to know whether to include the vector graphics part for (some of) his graphics or not, font changes sometimes require some knowledge of the internal commands of the LaTeX part (depending on the tool that generated the graphics), and the user is not intended to change the scaling of just the vector graphics part without affecting the text format. Furthermore, rotating, scaling and resizing the whole graphics requires extra-macros, e. g. `\rotatebox`, `\scalebox`, or `\resizebox` from the graphics/graphicx package.

The intention of this package is to provide a macro for easier handling of combined EPS/LaTeX or PDF/LaTeX graphics, just as easy as `\includegraphics`, and removing the limitations mentioned above.

2 Usage

2.1 The macro

`\includecombinedgraphics` The macro `\includecombinedgraphics[`⟨*option list*⟩`]{`⟨*graphics file*⟩`}` is used to include a combined EPS/LaTeX or PDF/LaTeX file, where ⟨*graphics file*⟩ is the name of the LaTeX part of the graphics (*without* `.tex` extension) and ⟨*option list*⟩ is an optional list of ⟨*key*⟩`=`⟨*value*⟩ pairs (see below).

2.2 Basic macro options

The following options are processed first. If an option is specified several times, the last appearance will overwrite the previous ones.

`textfont`=⟨*value*⟩	one or more commands for reformatting the text in LaTeX part (e. g., `textfont=\Large\bfseries` for large boldface font or `textfont={}` for default font of graphics overwriting package option)
`textcolor`=⟨*value*⟩	one or more commands for recoloring the text in LaTeX part (e. g., `textcolor=\color{red}` for red text or `textcolor={}` for default text color of graphics overwriting package option)
`vecscale`=⟨*value*⟩	scaling factor for rescaling the graphics without affecting the font of the LaTeX part (Note that this will overwrite `vecwidth` or `vecheight`, if they are specified before.)
`vecwidth`=⟨*value*⟩	analogous to `vecscale`, but the scaling factor will be calculated such that the new width of the total graphics will be ⟨*value*⟩; the aspect ratio will be preserved (Note that this will overwrite `vecscale` or `vecheight`, if they are specified before.)
`vecheight`=⟨*value*⟩	analogous to `vecscale`, but the scaling factor will be calculated such that the new (total) height of the total graphics will be ⟨*value*⟩; the aspect ratio will be preserved (Note that this will overwrite `vecscale` or `vecwidth`, if they are specified before.)
`vecinclude`=⟨*value*⟩	generate code to include the vector graphics part; ⟨*value*⟩s are: `auto` to automatically determine if code is required (default), `true`/`false` to always/never generate

	that code, or `overwrite` to disable any inclusion of a vector graphics part in the LaTeX part and generate new code
`vecfirst`=⟨*value*⟩	inclusion order of vector graphics part; valid ⟨*value*⟩s are: `true` (default) or `false` to include the vector graphics part before or after the LaTeX part, respectively
`vecfile`=⟨*value*⟩	filename of vector graphics part, if different from filename of LaTeX part (except for extension); implies `vecinclude=overwrite`

Note: If the vector graphics part is already included by the LaTeX part, `vecinclude` and `vecscale` will require the vector graphics part to be included by an `\includegraphics` command for proper functioning. Other graphics inclusion macros will not be detected. Furthermore, `vecscale` also requires that the LaTeX part puts its text inside a `picture` environment whose scaling is set with `\setlength{\unitlength}{`⟨*some length*⟩`}`. These two prerequisites are fulfilled by graphics exported from Gnuplot and Xfig.

2.3 Extended macro options

The following options are processed *after* the basic options *in the order* of their appearances and affect the *whole* graphics (LaTeX and vector graphics part). They may be specified several times.

Rotating

Rotating is basically performed with the means of `\rotatebox` from the graphics/graphicx package.

`angle`=⟨*value*⟩	angle of rotation

The following options can be used to fine adjust the effect of a rotation and apply to only the next `angle` value specified in the options list.

`origin`=⟨*value*⟩	up to two of `lrctbB` (`B` stands for the baseline)
`x`=⟨*value*⟩	the `x` coordinate of the center of rotation
`y`=⟨*value*⟩	the `y` coordinate of the center of rotation
`units`=⟨*value*⟩	units of rotation angle (e. g., `-360` denotes clockwise rotation)

Rescaling

Rescaling is performed with the means of `\scalebox` from the graphics package.

`scale`=⟨*value*⟩	scaling factor
`hscale`=⟨*value*⟩	horizontal scaling factor
`vscale`=⟨*value*⟩	vertical scaling factor

Note: A scaling via `hscale` and `vscale` will only be performed, if a pair of both values is specified.

Resizing

Resizing is performed with the means of `\resizebox` from the graphics package.

`height`=⟨*value*⟩	height of graphics
`totalheight`=⟨*value*⟩	height + depth of graphics
`width`=⟨*value*⟩	width of graphics
`keepaspectratio`	if only one of the values for `height`/`totalheight` or `width` is specified, the other one will be calculated to keep the aspect ratio

Note: Resizing will only be performed, if a pair of `height`/`totalheight` and `width` is specified or one of them is specified followed by `keepaspectratio`.

2.4 Package options

The following options can be passed to the package as defaults for the macro options. If an option is specified several times, the last appearance will overwrite the previous ones.

`textfontcmd`=⟨*value*⟩	name of a command (*without* leading backslash) for reformatting text in LaTeX part
`textcolorcmd`=⟨*value*⟩	name of a command (*without* leading backslash) for recoloring text in LaTeX part
`vecscale`=⟨*value*⟩	see section 2.2
`vecwidth`=⟨*value*⟩	see section 2.2; in contrast to the macro options, you must not use a length as ⟨*value*⟩ here (workaround: use `\the\somelength`)
`vecheight`=⟨*value*⟩	see section 2.2; in contrast to the macro options, you must not use a length as ⟨*value*⟩ here (workaround: use `\the\somelength`)
`vecinclude`=⟨*value*⟩	see section 2.2
`vecfirst`=⟨*value*⟩	see section 2.2

3 Bugs, problems, and suggestions

Please report bugs and problems or send suggestions for this package to Christian Schneider. Check for updates before reporting bugs at the website mentioned above.

4 Implementation

Load required packages for graphics, color and ⟨*key*⟩=⟨*value*⟩ pairs:

```
1 \RequirePackage{keyval}
2 \RequirePackage{graphicx}
3 \RequirePackage{color}
```

4.1 Package options

This `if` is required to remember the inclusion order. The macro specifies the package option from `vecscale`, `vecwidth`, `vecheight` used last.

```
\newif\ifcgr@vecfirst
\def\cgr@vecscale@type{0}
```

Definitions of package options as ⟨*key*⟩=⟨*value*⟩ pairs. The ⟨*value*⟩s are saved in the specified macros that are undefined by default.

```
\newcommand*\cgr@defopts{\define@key{cgr}}
\newcommand*\cgr@setopts{\setkeys{cgr}}
\cgr@defopts{textfontcmd}{\gdef\cgr@textfont@default{\@nameuse{#1}}}
\cgr@defopts{textcolorcmd}{\gdef\cgr@textcolor@default{\@nameuse{#1}}}
\cgr@defopts{vecscale}{\gdef\cgr@vecscale@default{#1}%
  \gdef\cgr@vecscale@type{0}}
\cgr@defopts{vecwidth}{\gdef\cgr@vecwidth@default{#1}%
  \gdef\cgr@vecscale@type{1}}
\cgr@defopts{vecheight}{\gdef\cgr@vecheight@default{#1}%
  \gdef\cgr@vecscale@type{2}}
\cgr@defopts{vecinclude}{\gdef\cgr@vecinclude@default{#1}}
\cgr@defopts{vecfirst}{\gdef\cgr@vecfirst@default{#1}}
```

Next the package options are processed.

```
\DeclareOption*{%
  \expandafter\cgr@setopts\expandafter{\CurrentOption}%
}
\ProcessOptions\relax
```

4.2 Basic macro options

Now the ⟨*key*⟩=⟨*value*⟩ pairs for the optional argument of `\includecombinedgraphics` are defined. The following two macro are just shortcuts for this purpose.

```
\newcommand*\cgr@defopts@combgrphcs{\define@key{cgr@combgrphcs}}
\newcommand*\cgr@setopts@combgrphcs{\setkeys{cgr@combgrphcs}}
```

The basic macro options set some macros that are executed while tinkering the graphics from its two parts.

`textfont` and `textcolor` set `\cgr@textfont` and `\cgr@textcolor` macro, respectively, that will be applied to the text in the LaTeX part of graphics *and* disable further formatting of fonts and colors inside the LaTeX part by overwriting some macros afterwards (if not empty).

```
\cgr@defopts@combgrphcs{textfont}{%
  \def\@tempa{#1}%
  \ifx\@tempa\@empty%
    \def\cgr@textfont{}%
  \else%
    \def\cgr@textfont{%
      #1%
      \def\SetFigFont{\cgr@gobblefive}%     %% Xfig <= 3.2.4
      \def\SetFigFontNFSS{\cgr@gobblefive}% %% Xfig >= 3.2.5-alpha
    }%
  \fi%
}
\cgr@defopts@combgrphcs{textcolor}{%
```

```
  \def\@tempa{#1}%
  \ifx\@tempa\@empty%
    \def\cgr@textcolor{}%
  \else%
    \def\cgr@textcolor{%
      #1%
      \def\color{\cgr@gobble@optone}%
    }%
  \fi%
}
```

`vecscale` sets the macro `\cgr@vecscale` to the scaling factor. Furthermore, it overwrites `\setlength` and `\includegraphics` in order to apply the scaling factor to the `picture` environment inside the LaTeX part (via `\setlength{\unitlength}...`) and to the `\includegraphics` command for inclusion of the vector graphics part, respectively.

```
\cgr@defopts@combgrphcs{vecscale}{%
  \def\cgr@vecscale{#1}%
  \def\setlength{\cgr@setlength}%
  \def\includegraphics{\cgr@includegraphics}%
}
```

`vecwidth` and `vecheight`, respectively, calculate the scaling factor for `vecscale` such that the width or (total) height of the vector graphics part equals the specified ⟨*dimen*⟩. `\vecscale` is set to `-1` to indicate that the factor shall be calculated.

```
\cgr@defopts@combgrphcs{vecwidth}{%
  \cgr@vecscale@wd{\cgr@texfile}{\cgr@vecfile}{#1}%
  \def\setlength{\cgr@setlength}%
  \def\includegraphics{\cgr@includegraphics}%
}
\cgr@defopts@combgrphcs{vecheight}{%
  \cgr@vecscale@ht{\cgr@texfile}{\cgr@vecfile}{#1}%
  \def\setlength{\cgr@setlength}%
  \def\includegraphics{\cgr@includegraphics}%
}
```

`vecinclude` switches between the different ⟨*value*⟩s by setting `\cgr@vecinclude` to one of the for macros `\cgr@requires@graphics`⟨*value*⟩.

```
\cgr@defopts@combgrphcs{vecinclude}{%
  \def\cgr@vecinclude{\@nameuse{cgr@requires@graphics#1}}%
}
```

`vecfirst` switches between the ⟨*value*⟩s `true` and `false` by setting `\cgr@vecfirsttrue` or `\cgr@vecfirstfalse`, respectively.

```
\cgr@defopts@combgrphcs{vecfirst}{%
  \@nameuse{cgr@vecfirst#1}%
}
```

`vecfile` resets `\cgr@vecfile` that defaults to the mandatory argument of `\includecombinedgraphics` to something else *and* sets `vecinclude` to `overwrite`.

```
\cgr@defopts@combgrphcs{vecfile}{%
  \def\cgr@vecfile{#1}%
  \cgr@setopts@combgrphcs{vecinclude=overwrite}%
}
```

4.3 Extended macro options

`\includecombinedgraphics` first tinkers the graphics applying the basic macro options and saves the result in `\cgr@curr@pic`. The extended macro options will redefine `\cgr@curr@pic` in the order of their occurance and put a `\rotatebox`, `\scalebox` or `\resizebox` around it, if all information is already available to do so: `\cgr@curr@pic` → `\...box{\cgr@curr@pic}`. Otherwise the ⟨*value*⟩ will be saved in a macro for later processing (e. g., if `hscale` is set without `vscale` being set before).

```
\cgr@defopts@combgrphcs{angle}{%
  \ifx\cgr@curr@scaleopts\@empty%
    \cgr@raddto@macro[groupfirst]{\cgr@curr@pic}{\rotatebox{#1}}%
  \else%
    \cgr@raddto@macro[groupfirst]{\cgr@curr@pic}{]}%
    \cgr@raddto@macro[expand]{\cgr@curr@pic}{\cgr@curr@scaleopts}%
    \cgr@raddto@macro{\cgr@curr@pic}{\rotatebox[}%
    \def\cgr@curr@scaleopts{}%
  \fi%
}
\cgr@defopts@combgrphcs{origin}{\cgr@addto@macro{\cgr@curr@scaleopts}{%
  origin=#1,}%
}
\cgr@defopts@combgrphcs{x}{\cgr@addto@macro{\cgr@curr@scaleopts}{x=#1,}}
\cgr@defopts@combgrphcs{y}{\cgr@addto@macro{\cgr@curr@scaleopts}{y=#1,}}
\cgr@defopts@combgrphcs{units}{\cgr@addto@macro{\cgr@curr@scaleopts}{units=#1,}}
\cgr@defopts@combgrphcs{scale}{%
  \cgr@raddto@macro[groupfirst]{\cgr@curr@pic}{\scalebox{#1}}%
}
\cgr@defopts@combgrphcs{hscale}{%
  \def\cgr@curr@hscale{#1}%
  \ifx\cgr@curr@vscale\@empty%
  \else%
    \cgr@raddto@macro[groupfirst]{\cgr@curr@pic}{]}%
    \cgr@raddto@macro[expand]{\cgr@curr@pic}{\cgr@curr@vscale}%
    \cgr@raddto@macro{\cgr@curr@pic}{[}%
    \cgr@raddto@macro[expand,groupsecond]{\cgr@curr@pic}{\cgr@curr@hscale}%
    \cgr@raddto@macro{\cgr@curr@pic}{\scalebox}%
    \def\cgr@curr@hscale{}%
    \def\cgr@curr@vscale{}%
  \fi%
}
\cgr@defopts@combgrphcs{vscale}{%
  \def\cgr@curr@vscale{#1}%
  \ifx\cgr@curr@hscale\@empty%
  \else%
    \cgr@raddto@macro[groupfirst]{\cgr@curr@pic}{]}%
    \cgr@raddto@macro[expand]{\cgr@curr@pic}{\cgr@curr@vscale}%
    \cgr@raddto@macro{\cgr@curr@pic}{[}%
    \cgr@raddto@macro[expand,groupsecond]{\cgr@curr@pic}{\cgr@curr@hscale}%
    \cgr@raddto@macro{\cgr@curr@pic}{\scalebox}%
    \def\cgr@curr@hscale{}%
    \def\cgr@curr@vscale{}%
  \fi%
```

```
}
\cgr@defopts@combgrphcs{height}{%
  \def\cgr@curr@height{#1}%
  \ifx\cgr@curr@width\@empty%
  \else%
    \cgr@raddto@macro[groupfirst,groupsecond,expand]{\cgr@curr@pic}{%
      \cgr@curr@height%
    }%
    \cgr@raddto@macro[groupsecond,expand]{\cgr@curr@pic}{\cgr@curr@width}%
    \cgr@raddto@macro{\cgr@curr@pic}{\resizebox}%
    \def\cgr@curr@height{}%
    \def\cgr@curr@width{}%
  \fi%
}
\cgr@defopts@combgrphcs{totalheight}{%
  \def\cgr@curr@totalheight{#1}%
  \ifx\cgr@curr@width\@empty%
  \else%
    \cgr@raddto@macro[groupfirst,groupsecond,expand]{\cgr@curr@pic}{%
      \cgr@curr@totalheight%
    }%
    \cgr@raddto@macro[groupsecond,expand]{\cgr@curr@pic}{\cgr@curr@width}%
    \cgr@raddto@macro{\cgr@curr@pic}{\resizebox*}%
    \def\cgr@curr@totalheight{}%
    \def\cgr@curr@width{}%
  \fi%
}
\cgr@defopts@combgrphcs{width}{%
  \def\cgr@curr@width{#1}%
  \ifx\cgr@curr@height\@empty%
    \ifx\cgr@curr@totalheight\@empty%
    \else%
      \cgr@raddto@macro[groupfirst,groupsecond,expand]{\cgr@curr@pic}{%
        \cgr@curr@totalheight%
      }%
      \cgr@raddto@macro[groupsecond,expand]{\cgr@curr@pic}{\cgr@curr@width}%
      \cgr@raddto@macro{\cgr@curr@pic}{\resizebox*}%
      \def\cgr@curr@totalheight{}%
      \def\cgr@curr@width{}%
    \fi%
  \else%
    \cgr@raddto@macro[groupfirst,groupsecond,expand]{\cgr@curr@pic}{%
      \cgr@curr@height%
    }%
    \cgr@raddto@macro[groupsecond,expand]{\cgr@curr@pic}{\cgr@curr@width}%
    \cgr@raddto@macro{\cgr@curr@pic}{\resizebox}%
    \def\cgr@curr@height{}%
    \def\cgr@curr@totalheight{}%
    \def\cgr@curr@width{}%
  \fi%
}
\cgr@defopts@combgrphcs{keepaspectratio}[]{%
  \ifx\cgr@curr@height\@empty%
    \ifx\cgr@curr@totalheight\@empty%
```

```
      \ifx\cgr@curr@width\@empty%
      \else%
        \cgr@setopts@combgrphcs{height=!}%
      \fi%
    \else%
      \cgr@setopts@combgrphcs{width=!}%
    \fi%
  \else%
    \cgr@setopts@combgrphcs{width=!}%
    \def\cgr@curr@totalheight{}%
  \fi%
}
```

4.4 The macro

First of all, `\cgr@curr@pic` and the `\cgr@requires@graphics...` macros are initialized.

```
\long\def\cgr@curr@pic{}
\newif\ifcgr@requires@graphics\cgr@requires@graphicstrue
\def\cgr@requires@graphicsauto{}
\def\cgr@requires@graphicsoverwrite{%
  \cgr@requires@graphicstrue%
  \def\includegraphics{\cgr@gobble@optone}%
}
```

Now the macros and `if` for the basic macro options and for temporarily saving ⟨*value*⟩s of the extended macro options are initialized.

```
\def\cgr@textfont{}
\def\cgr@textcolor{}
\def\cgr@vecscale{1}
\def\cgr@vecinclude{\cgr@requires@graphicsauto}
\cgr@vecfirsttrue
\def\cgr@texfile{}
\def\cgr@vecfile{}
\def\cgr@curr@hscale{}
\def\cgr@curr@vscale{}
\def\cgr@curr@scaleopts{}
\def\cgr@curr@height{}
\def\cgr@curr@width{}
\def\cgr@curr@totalheight{}
```

At this point the macro itself is defined.

```
\newcommand{\includecombinedgraphics}[2][]{%
  \begingroup%
```

To get to know, whether the vector graphics are already included in the LaTeX parts or not, a check for an appearance of `\includegraphics` in the LaTeX part is performed: the LaTeX parts are expanded inside a box with `\includegraphics` being redefined to set an appropriate conditional (and eat the arguments of `\includegraphics`).

```
    \global\cgr@requires@graphicstrue%
    \setbox\@tempboxa\hbox{%
      \def\includegraphics{%
        \global\cgr@requires@graphicsfalse\cgr@gobble@optone%
```

```
    }%
    \input{#2}%
  }%
```

Now the the macros resulting from the basic macro options, the vector graphics part and the LaTeX part are added to the (empty) `\cgr@curr@pic` macro. We have to distinguish between two cases: (1) If the vector graphics part is included before the LaTeX part (`vecfirst=true`), the vector graphics file will simply be loaded by `\includegraphics` inside a picture environment followed by the LaTeX part included by `\input`. (2) If the vector graphics part is included after the LaTeX part (`vecfirst=false`), the `\includegraphics` macro will be put inside the `picture` environment of the LaTeX part at the offset position passed to the `picture` environment. Therefore, we will have to wrap the `\picture` macro to gain access to the offsets passed to the `picture` environment in the LaTeX part and wrap the `\endpicture` macro to add `\includegraphics` into this `picture` environment.

```
  \cgr@addto@macro{\cgr@curr@pic}{%
    %% from basic macro options
    \cgr@vecinclude\cgr@textfont\cgr@textcolor%
    %% inclusion of vector graphics part
    \ifcgr@requires@graphics%
      \ifcgr@vecfirst%
        \begin{picture}(0,0)%
          \cgr@includegraphics@orig[scale=\cgr@vecscale]{\cgr@vecfile}%
        \end{picture}%
      \else%
        \def\picture{\cgr@picture}%
        \def\endpicture{%
          \put(\cgr@picture@xoffs,\cgr@picture@yoffs)%
            {\cgr@includegraphics@orig[scale=\cgr@vecscale]{\cgr@vecfile}}%
          \cgr@endpicture@orig%
        }%
      \fi%
    \fi%
    %% inclusion of \LaTeX{} part
    \input{#2}%
  }%
```

Afterwards, a macro containing the name of the LaTeX and vector graphics part, respectively, is initialized.

```
  \def\cgr@vecfile{#2}%
  \def\cgr@texfile{#2}%
```

If set, the package options will be processed. (Suggestions for easier handling of package options are welcome.)

```
  \if0\cgr@vecscale@type%
    \@ifundefined{cgr@vecscale@default}{}{%
      \cgr@setopts@combgrphcs{vecscale=\cgr@vecscale@default}%
    }%
  \fi%
  \if1\cgr@vecscale@type%
    \@ifundefined{cgr@vecwidth@default}{}{%
      \cgr@setopts@combgrphcs{vecwidth=\cgr@vecwidth@default}%
    }%
```

```
    \fi%
    \if2\cgr@vecscale@type%
      \@ifundefined{cgr@vecheight@default}{}{%
        \cgr@setopts@combgrphcs{vecheight=\cgr@vecheight@default}%
      }%
    \fi%
    \@ifundefined{cgr@textfont@default}{}{%
      \cgr@setopts@combgrphcs{textfont=\cgr@textfont@default}%
    }%
    \@ifundefined{cgr@textcolor@default}{}{%
      \cgr@setopts@combgrphcs{textcolor=\cgr@textcolor@default}%
    }%
    \@ifundefined{cgr@vecinclude@default}{}{%
      \cgr@setopts@combgrphcs{vecinclude=\cgr@vecinclude@default}%
    }%
    \@ifundefined{cgr@vecfirst@default}{}{%
      \cgr@setopts@combgrphcs{vecfirst=\cgr@vecfirst@default}%
    }%
```

Now, the options passed the the macro are processed.

```
    \cgr@setopts@combgrphcs{#1}%
```

Finally, the macro `\cgr@curr@pic` is complete and can be output.

```
    \cgr@curr@pic%
  \endgroup%
}
```

4.5 Helper macros

This macro eats five arguments (analogous to `\@gobble` or `\@gobbletwo` from the LaTeX kernel).

```
\long\def\cgr@gobblefive#1#2#3#4#5{}
```

This macro eats all stars (if any), all arguments in square brackets (if any) and one mandatory argument.

```
\long\def\cgr@gobble@optone{\@ifstar{\cgr@gobble@optone}{\cgr@gobble@optone@}}
\long\def\cgr@gobble@optone@{\@ifnextchar [{\cgr@gobble@optone@@}{\@gobble}}
\long\def\cgr@gobble@optone@@[#1]{\cgr@gobble@optone@}
```

In order to rescale the `picture` environment inside the LaTeX part, the `\setlength{\unitlength}{`⟨*some dimen*⟩`}` must be changed to scale ⟨*some dimen*⟩ by `\cgr@vecscale` at its first appearance in the LaTeX part. This is the `\setlength` substitute to do so.

```
\let\cgr@setlength@orig=\setlength
\def\cgr@setlength#1#2{%
  \ifx#1\unitlength%
    \@tempdima=#2%
    \cgr@setlength@orig{#1}{\cgr@vecscale\@tempdima}%
    \def\setlength{\cgr@setlength@orig}%
  \else%
    \cgr@setlength@orig{#1}{#2}%
  \fi%
}
```

Additionally, a `scale=\cgr@vecscale` option must be passed to the `\includegraphics` macro inside the LaTeX part. This is the substitute of `\includegraphics` for this purpose.

```
\let\cgr@includegraphics@orig=\includegraphics
\def\cgr@includegraphics{%
  \@ifstar{%
    \cgr@includegraphics@s@%
  }{%
    \cgr@includegraphics@%
  }%
}
\def\cgr@includegraphics@{%
  \@ifnextchar [{%
    \cgr@includegraphics@@%
  }{%
    \cgr@includegraphics@orig[scale=\cgr@vecscale]%
  }%
}
\def\cgr@includegraphics@@[#1]{%
  \@ifnextchar [{%
    \cgr@includegraphics@@@[#1]%
  }{%
    \cgr@includegraphics@orig[#1,scale=\cgr@vecscale]%
  }%
}
\def\cgr@includegraphics@@@[#1][#2]{%
  \cgr@includegraphics@orig[#1][#2,scale=\cgr@vecscale]%
}
\def\cgr@includegraphics@s@{%
  \@ifnextchar [{%
    \cgr@includegraphics@s@@%
  }{%
    \cgr@includegraphics@orig*[scale=\cgr@vecscale]%
  }%
}
\def\cgr@includegraphics@s@@[#1]{%
  \@ifnextchar [{%
    \cgr@includegraphics@s@@@[#1]%
  }{%
    \cgr@includegraphics@orig*[#1,scale=\cgr@vecscale]%
  }%
}
\def\cgr@includegraphics@s@@@[#1][#2]{%
  \cgr@includegraphics@orig*[#1][#2,scale=\cgr@vecscale]%
}
```

To play the trick of including the vector graphics part after the LaTeX part (`vecfirst=false`), we need to know the offset passed to the picture environment in the LaTeX part. Therefore, the `\picture` macro is redefined.

```
\let\cgr@picture@orig=\picture
\let\cgr@endpicture@orig=\endpicture
\long\def\cgr@picture(#1,#2){%
  \@ifnextchar({%
    \cgr@picture@(#1,#2)%
```

```
  }{%
    \cgr@picture@(#1,#2)(0,0)%
  }%
}
\def\cgr@picture@(#1,#2)(#3,#4){%
  \def\cgr@picture@xoffs{#3}%
  \def\cgr@picture@yoffs{#4}%
  \cgr@picture@orig(#1,#2)(#3,#4)%
}
```

This macro is identical to `\l@addto@macro` from koma-script bundle. It adds the stuff passed to its second argument to the end of the macro from its first argument.

```
\newcommand{\cgr@addto@macro}[2]{%
  \begingroup\toks@\expandafter{#1#2}%
    \edef\@tempa{\endgroup\def\noexpand#1{\the\toks@}}%
  \@tempa%
}
```

The next macro is similar, but adds the stuff passed to its second argument to the *beginning* of the macro from its first argument. An optional argument allows for fine tuning: A comma-separated list containing `expand` (expands the stuff from the second argument before adding it), `groupfirst`, and/or `groupsecond` (puts the stuff from the first/second argument in braces before adding) may be passed.

```
\newcommand*\cgr@defopts@raddto{\define@key{cgr@raddto}}
\newcommand*\cgr@setopts@raddto{\setkeys{cgr@raddto}}
\newif\ifcgr@raddto@expand\cgr@raddto@expandfalse
\newif\ifcgr@raddto@groupfirst\cgr@raddto@groupfirstfalse
\newif\ifcgr@raddto@groupsecond\cgr@raddto@groupsecondfalse
\cgr@defopts@raddto{expand}[true]{\@nameuse{cgr@raddto@expand#1}}
\cgr@defopts@raddto{groupfirst}[true]{\@nameuse{cgr@raddto@groupfirst#1}}
\cgr@defopts@raddto{groupsecond}[true]{\@nameuse{cgr@raddto@groupsecond#1}}
\newtoks\cgr@token@a
\newtoks\cgr@token@b
\newcommand{\cgr@raddto@macro}[3][]{%
  \begingroup%
    \cgr@setopts@raddto{#1}%
    \cgr@token@a\expandafter{#2}%
    \ifcgr@raddto@expand%
      \cgr@token@b\expandafter{#3}%
    \else%
      \cgr@token@b{#3}%
    \fi%
    \ifcgr@raddto@groupfirst%
      \ifcgr@raddto@groupsecond%
        \edef\@tempa{\endgroup%
          \def\noexpand#2{{\the\cgr@token@b}{\the\cgr@token@a}}%
        }%
      \else%
        \edef\@tempa{\endgroup%
          \def\noexpand#2{\the\cgr@token@b{\the\cgr@token@a}}%
        }%
      \fi%
    \else%
```

```
      \ifcgr@raddto@groupsecond%
        \edef\@tempa{\endgroup%
          \def\noexpand#2{{\the\cgr@token@b}\the\cgr@token@a}%
        }%
      \else%
        \edef\@tempa{\endgroup%
          \def\noexpand#2{\the\cgr@token@b\the\cgr@token@a}%
        }%
      \fi%
    \fi%
  \@tempa%
}
```

The way of calculating the scaling factor for `vecwidth` and `vecheight` is a simplified and adapted version of `\ratio` from the `calc.sty` package. First, the width or (total) height of the original graphics part is measured. In order to do this, the graphics are put into a box. Afterwards, the places before the decimal point are calculated by TEX's built-in integer division. Finally, the decimal places are calculated with an algorithm as division has been taught at school.

```
\newcount\cgr@tempdigit
\newcommand*\cgr@vecscale@wd[3]{%
  \cgr@vecscale@graphicsbox{#1}{#2}%
  \cgr@vecscale@calc{\wd\@tempboxa}{#3}%
}
\newcommand*\cgr@vecscale@ht[3]{%
  \cgr@vecscale@graphicsbox{#1}{#2}%
  \@tempdimc=\ht\@tempboxa%
  \advance\@tempdimc by\dp\@tempboxa%
  \cgr@vecscale@calc{\@tempdimc}{#3}%
}
\newcommand*\cgr@vecscale@graphicsbox[2]{%
  \setbox\@tempboxa\hbox{%
    \begingroup%
      \def\setlength{\cgr@setlength@orig}%
      \def\includegraphics{\cgr@gobble@optone}%
      \input{#1}%
    \endgroup%
  }%
}
\newcommand*\cgr@vecscale@calc[2]{%
  \@tempdimb=#1%
  \@tempcntb=\@tempdimb%
  \@tempdima=#2%
  \@tempcnta=\@tempdima%
  \cgr@tempdigit=\@tempcnta%
  \divide\cgr@tempdigit by\@tempcntb%
  \edef\cgr@vecscale{\the\cgr@tempdigit.}%
  \cgr@next@digit\cgr@next@digit\cgr@next@digit%
  \cgr@next@digit\cgr@next@digit\cgr@next@digit%
}
\newcommand*\cgr@next@digit{%
  \multiply\cgr@tempdigit by\@tempcntb%
  \advance\@tempcnta by-\cgr@tempdigit%
  \multiply\@tempcnta by10%
```

```
417   \cgr@tempdigit=\@tempcnta%
418   \divide\cgr@tempdigit by\@tempcntb%
419   \edef\cgr@vecscale{\cgr@vecscale\the\cgr@tempdigit}%
420 }
```

Change History

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.

D

E

F

G

H

I