The reflectgraphics package

Oliver Reiche
oliver.reiche@gmail.com

v0.2c from 2015/07/07

Abstract

This small package provides a new macro, which adds fancy reflections at the bottom of graphics. To keep things simple, this new macro can be used in a similar way to the `\includegraphics` macro of the graphicx package.

1 Introduction

Having a fading reflection or mirroring effect at the bottom of graphics can dramatically increase the fancyness of documents and in particular of presentations. Such an effect can be achieved by creating the reflection beforehand with an image editing software or by using the tikz package. Using Ti*k*Z can be very tricky and many users might not be familiar with describing graphics by writing code. In particular when applying image manipulation operations (like scaling, rotating, trimming) on graphics, handling Ti*k*Z code and positioning nodes correctly can be challenging.

To remove this obstacle for inexperienced users and allow them to easily include graphics with fancy reflections, this package offers a simple alternative. By just specifying a graphics file, the macros provided by this package will render it and its reflection automatically.

2 Usage

No big surprise, the package is loaded using:

`\usepackage[`⟨*options*⟩`]{reflectgraphics}`

The options specified as ⟨*options*⟩ are globally set for the whole document and can be obliterated by locally defined options. A description of valid options can be found in subsection 2.2.

2.1 Macros

The only new macro provided by this package is:

`\reflectgraphics` `\reflectgraphics[`⟨*options*⟩`]{`⟨*path*⟩`}`

It can be used to render graphics together with their reflections at the bottom, which are fading from a particular starting opacity to fully transparent. The dimensions of a graphic and the properties of its reflection can be defined by ⟨*options*⟩ (see subsection 2.2).

The mandatory argument ⟨*path*⟩ must contain a valid path to a graphics file of a supported file type. This package internally uses `\includegraphics`. Therefore, basically every graphics type that is supported by `\includegraphics` is also supported by this package. The file extension may be omitted in ⟨*path*⟩. To see some examples have a look at subsection 2.3.

2.2 Options

Options can be specified globally with the `\usepackage` statement or locally for each `\reflectgraphics` statement. Local options override values of global options. There are two types of options: the ones that affect graphics and the ones that affect only the rendering of reflections.

graphics The following options describe the properties of graphics.

`width` option
`height` option
To set the dimensions of the graphic to render, the options `width`=⟨*dim*⟩ and `height`=⟨*dim*⟩ can be specified. If only one of these options is specified, the original aspect ratio will be left untouched. Values passed to these options must be typical TEX dimensions with units like `cm`, `em`, `pt`, and so on.

`scale` option
Instead of a manually defining graphics dimensions, the option `scale`=⟨*float*⟩ can be set as well. Hereby the original aspect ratio and size will be chosen scaled by the provided floating point value. Valid values are all values that are greater than 0. If either `width` or `height` is specified, the `scale` option will be ignored.

`angle` option
The graphics and corresponding reflections can be rotated by specifying the option `angle`=⟨*float*⟩. This floating point value describes the rotation angle in degrees. Valid angle values can be positive as well as negative.

`trim` option
Similar to `\includegraphics` trimming and clipping can be applied to render only subregions of graphics. This can be done by defining the option `trim`=⟨*dim*⟩ ⟨*dim*⟩ ⟨*dim*⟩ ⟨*dim*⟩. The option takes four dimension arguments that define the trimming in the following order: *left bottom right top*. If the dimension unit is omitted than `bp` is used by default, which is similar to `\includegraphics`. Unlike `\includegraphics` the `clip` option does not have to be specified.

reflections The following options describe the properties of graphics reflections.

`distance` option
The distance between the actual graphic and its reflection can be defined by setting the option `distance`=⟨*dim*⟩. If it is not set, a default value of 3.0 `pt` will be used. The specified value must be a TEX dimension.

`length` option
To set the size of the graphics reflection, specify the option `length`=⟨*float*⟩.

This option defines a percentage value, which represents the length (i. e. the height) of the reflection with respect to the height of the corresponding graphic. Its value is a floating point number and must lie within the interval $[0, 1]$. The default value is set to 0.5, which results in a reflection height that is equal to 50 % of the graphics height.

`opacity` option

Changing the starting opacity for the fading of the reflection influences the impression how *strong* the reflection is. The opacity can be modified by setting the `opacity`=⟨*float*⟩ option. Its value is a floating point number within the interval $[0, 1]$. The default value is 0.5, which equates to 50 % opacity.

2.3 Examples

The simplest way to use `\reflectgraphics` is to omit all optional arguments and let the underlaying `\includegraphics` macro choose the dimensions of the graphic:

```
\reflectgraphics{example.jpg}
```

To change the graphics size in a way whereat the aspect ratio should be kept as it is, define either `width`, `height` or use the `scale` option:

```
\reflectgraphics[width=4cm]{example.jpg}
\reflectgraphics[height=4cm]{example.jpg}
\reflectgraphics[scale=0.25]{example.jpg}
```

If a specific aspect ratio should be forced, both dimensions have to be specified explicitly:

```
\reflectgraphics[width=4cm, height=2cm]{example.jpg}
```

Rotating graphics can be enabled by specifying the `angle` option. Note that rotation will increase the rectangular space (bounding box) that is taken by the graphic and its reflection. In particular the overall height can expand drastically, because the increased height of both the graphic and the reflection add up.

```
\reflectgraphics[width=1.5cm, angle=45]{example.jpg}
```

To trim and clip graphics together with the corresponding reflection (e. g. to highlight the olives only), one can straightforward proceed the same way as using `\includegraphics`, except that the `clip` option can be omitted:

```
\reflectgraphics[trim=115 60 80 100]{example.jpg}
```

In order to get a stronger and more crisp reflection, the options `opacity` and `length` can be set. For instance, to get a stronger starting opacity of 75 % and a

much shorter reflection length of only 20 % of the graphics height, set the following values:

```
\reflectgraphics[opacity=0.75, length=0.2]{example.jpg}
```

To automatically modify the distance between a graphic and its reflection when dimensions change, the distance and the graphics height can be set in relation to each other. The following code renders graphics with a height of 4 `cm` and sets the distance of the reflection to 10 % of the graphics height ($0.4\,\texttt{cm} = 0.1 \cdot 4\,\texttt{cm}$):

```
\newlength{\len}
\setlength{\len}{4cm}
\reflectgraphics[height=\len, distance=0.1\len]{example.jpg}
```

3 Implementation

```
\RequirePackage{kvoptions, keyval, graphicx, calc, tikz}
\usetikzlibrary{fadings}
\makeatletter
```

Holds the debug condition provided by option `debug`.

```
\newif\ifrg@debug
```

Holds the width length provided by option `width`.

```
\newlength\rg@width
```

Holds the height length provided by option `height`.

```
\newlength\rg@height
```

Holds the distance length provided by option `distance`.

```
\newlength\rg@distance
```

`\rg@scale` Holds the scale factor provided by option `scale`.

```
\def\rg@scale{}
```

`\rg@angle` Holds the angle value provided by option `angle`.

```
\def\rg@angle{}
```

`\rg@length` Holds the length factor for the reflection provided by option `length`.

```
\def\rg@length{}
```

`\rg@opacity` Holds the opacity factor for starting the reflection provided by option `opacity`.

```
\def\rg@opacity{}
```

Hold the trim lengths provided by option `trim`:
Left trim length

```
\newlength\rg@triml
```

Bottom trim length

```
\newlength\rg@trimb
```

Right trim length

```
\newlength\rg@trimr
```

Top trim length

```
\newlength\rg@trimt
```

Dimension holding the default unit for converting dimensions.

```
\newdimen\rg@unit \rg@unit=1bp%
```

Dimension temporarily used for storing converted dimensions.

```
\newdimen\rg@trim@dim%
```

`\rg@convtodim` Macro for converting an argument without unit to dimension with default unit.

```
\def\rg@convtodim#1#2{%
```

Set unit variable to default unit.

```
  \let\rg@unit@cur\rg@unit%
```

Macro for clearing the unit variable.

```
\def\rg@unit@off{\let\rg@unit@cur\relax}%
```

Clear unit variable after next assignment.

```
\afterassignment\rg@unit@off%
```

Assign argument #2 to dimension.

```
\rg@trim@dim#2\rg@unit@cur%
```

Copy resulting dimension to argument #1.

```
\setlength#1{\the\rg@trim@dim}%
}
```

`\rg@trim@parse` Macro for parsing trim values from single argument to `\rg@trim[lbrt]` and convert to dimension if no unit is specified.

```
\def\rg@trim@parse#1 #2 #3 #4 #5\\{%
  \rg@convtodim{\rg@triml}{#1}%
  \rg@convtodim{\rg@trimb}{#2}%
  \rg@convtodim{\rg@trimr}{#3}%
  \rg@convtodim{\rg@trimt}{#4}%
}
```

`\rg@checkval` Macro for checking whether value is in range or not.

```
\def\rg@checkval#1#2#3#4{%
  \ifdim#1pt<#3pt%
    \PackageError{reflectgraphics}{%
      The specified #2 value '#1' is less than #3%
    }{%
      Specify the #2 value within the interval [#3,#4].%
    }%
  \else\ifdim#1pt>#4pt%
    \PackageError{reflectgraphics}{%
      The specified #2 value '#1' is greater than #4%
    }{%
      Specify the #2 value within the interval [#3,#4].%
    }%
  \fi\fi%
}
```

`debug` key Sets debug condition.

```
\define@key{reflectgraphics}{debug}[true]{%
  \csname rg@debug\ifx\relax#1\relax true\else#1\fi\endcsname}
```

`width` key Sets width length.

```
\define@key{reflectgraphics}{width}{%
  \setlength\rg@width{#1}}
```

`height` key Sets height length.

```
\define@key{reflectgraphics}{height}{%
  \setlength\rg@height{#1}}
```

`scale` key Sets scale factor.

```
\define@key{reflectgraphics}{scale}{%
  \def\rg@scale{#1}}
```

`angle` key Sets angle factor.

```
\define@key{reflectgraphics}{angle}{%
  \def\rg@angle{#1}}
```

`trim` key Sets trim lengths.

```
\define@key{reflectgraphics}{trim}{%
  \rg@trim@parse#1 \\}
```

`clip` key Accept but ignore option `clip`.

```
\define@key{reflectgraphics}{clip}[true]{}
```

`distance` key Sets reflection distance length.

```
\define@key{reflectgraphics}{distance}{%
  \setlength\rg@distance{#1}}
```

`length` key Sets reflection length factor after checking that value is in range.

```
\define@key{reflectgraphics}{length}{%
  \rg@checkval{#1}{length}{0}{1}%
  \def\rg@length{#1}}
```

`opacity` key Sets reflection starting opacity after checking that value is in range.

```
\define@key{reflectgraphics}{opacity}{%
  \rg@checkval{#1}{opacity}{0}{1}%
  \def\rg@opacity{#1}}
```

Set default values.

```
\setkeys{reflectgraphics}{%
  debug=false,%
  width=0pt,%
  height=0pt,%
  scale=1.0,%
  angle=0,%
  trim=0 0 0 0,%
  distance=3pt,%
  length=0.5,%
  opacity=0.5%
}
```

Process package options by defined keys.

```
\ProcessKeyvalOptions*
```

Holds width of the graphic including extra space when rotated.

```
\newlength{\rg@grp@width}
```

Holds height of the graphic including extra space when rotated.

```
\newlength{\rg@grp@height}
```

Holds height of the reflection including extra space when rotated.

```
\newlength{\rg@ref@height}
```

Holds overall height, which is used for clipping.

```
\newlength{\rg@clp@height}
```

Hold y offset of graphics node, which is used for positioning.

```
\newlength{\rg@grp@offset}
```

Hold y offset of reflections node, which is used for positioning.

```
\newlength{\rg@ref@offset}
```

`\reflectgraphics` This is the actual macro doing all the work.

```
\newcommand{\reflectgraphics}[2][]{%
```

Open new group so global options will not be overridden.

```
  \begingroup%
```

Load local options from first argument.

```
  \setkeys{reflectgraphics}{#1}%
```

Determine graphics dimensions. Use `\includegraphics` in `draft` mode to do that. If both the `width` and `height` options are specified, there is nothing that needs to be done here.

```
  \ifdim\rg@width=0pt\ifdim\rg@height=0pt
    \settowidth{\rg@width}{%
      \includegraphics[%
        draft,%
        scale=\rg@scale,%
        trim={\rg@triml} {\rg@trimb} {\rg@trimr} {\rg@trimt}]{#2}}%
    \settoheight{\rg@height}{%
      \includegraphics[%
        draft,%
        scale=\rg@scale,%
        trim={\rg@triml} {\rg@trimb} {\rg@trimr} {\rg@trimt}]{#2}}%
  \else% if width=0 && height!=0
    \settowidth{\rg@width}{%
      \includegraphics[%
        draft,%
        height=\rg@height,%
        trim={\rg@triml} {\rg@trimb} {\rg@trimr} {\rg@trimt}]{#2}}%
  \fi\else\ifdim\rg@height=0pt% && width!=0
    \settoheight{\rg@height}{%
      \includegraphics[%
        draft,%
        width=\rg@width,%
        trim={\rg@triml} {\rg@trimb} {\rg@trimr} {\rg@trimt}]{#2}}%
  \fi\fi%
```

Determine dimensions of the graphics bounding box if rotation is enabled. Again, `\includegraphics` in `draft` mode could be used to calculate the boundary size, but calling `\heightof` results in strange values for `angle` < 0 and `angle` > 90. Therefore, computing the boundary size is done manually using pgfmath, which turns out to be a little bit slower.

```
\ifdim\rg@angle pt=0pt%
  \setlength{\rg@grp@width}{\rg@width}%
  \setlength{\rg@grp@height}{\rg@height}%
\else%
```

This is totally strange: A simple division gets completely messed up if dividing a length by a larger length. So I need to take care to divide lengths only by smaller ones.

```
  \ifdim\rg@height>\rg@width%
    \pgfmathsetmacro{\rg@alpha}{atan(\rg@height/\rg@width)}%
  \else%
    \pgfmathsetmacro{\rg@alpha}{atan(1/(\rg@width/\rg@height)}%
  \fi%
  \pgfmathsetmacro{\rg@corner}{abs(cos(\rg@alpha))}%
  \pgfmathsetmacro{\rg@cornera}{abs(cos(\rg@alpha+\rg@angle))}%
  \pgfmathsetmacro{\rg@cornerb}{abs(cos(180-\rg@alpha+\rg@angle))}%
  \pgfmathsetmacro{\rg@scale@x}{%
    max(\rg@cornera/\rg@corner,\rg@cornerb/\rg@corner)}%
  \pgfmathsetmacro{\rg@corner}{abs(sin(\rg@alpha))}%
  \pgfmathsetmacro{\rg@cornera}{abs(sin(\rg@alpha+\rg@angle))}%
  \pgfmathsetmacro{\rg@cornerb}{abs(sin(180-\rg@alpha+\rg@angle))}%
  \pgfmathsetmacro{\rg@scale@y}{%
    max(\rg@cornera/\rg@corner,\rg@cornerb/\rg@corner)}%
  \setlength{\rg@grp@width}{\rg@scale@x\rg@width}%
  \setlength{\rg@grp@height}{\rg@scale@y\rg@height}%
\fi%
```

Compute height of reflection.

```
\setlength{\rg@ref@height}{%
  \rg@length\rg@grp@height}%
```

Compute y offset of the reflection that is used for node positioning. The center of the reflection node is moved to the origin of the coordinate system. This is done, because it seems that fading with a specific size can only be applied in a reasonable way at the origin (with option `fit fading` set to *false*).

```
\setlength{\rg@ref@offset}{%
  0.5\rg@grp@height-0.5\rg@ref@height}%
```

Compute y offset of the graphics node.

```
\setlength{\rg@grp@offset}{%
  0.5\rg@ref@height+\rg@distance+0.5\rg@grp@height}%
```

Compute overall height, which is used for clipping.

```
\setlength{\rg@clp@height}{%
  \rg@grp@height+\rg@distance+\rg@ref@height}%
```

The following `\tikzfadingfrompicture` macro does not work with Ti*k*Z externalize. Therefore, if the Ti*k*Z library external is loaded it must be disabled for the whole rendering.

```
\ifdefined\tikzexternaldisable%
  \tikzexternaldisable%
\fi%
```

Define custom fading `fade south`, which starts at 0 % transparency and ends at 100 %. The height of this fading equates to the height of the reflection. `\tikzfadingfrompicture` is used instead of `\tikzfading` because of its possibility to define shades with specific sizes. By doing so, this fading is not stretched across the whole node and therefore it can be used to hide a specific part of the reflection node.

```
\begin{tikzfadingfrompicture}[name=fade south]%
  \shade[top color=transparent!0, bottom color=transparent!100]%
    (0,0) rectangle (\rg@grp@width,\rg@ref@height);%
\end{tikzfadingfrompicture}%
```

Start the actual rendering.

```
\begin{tikzpicture}%
```

Draw help lines if `debug` option is specified.

```
  \ifrg@debug%
    \def\rg@padding{0.25\rg@grp@height}%
    \draw[help lines]%
      (-0.5\rg@grp@width-\rg@padding,%
        -0.5\rg@ref@height-\rg@distance-\rg@grp@height-\rg@padding)%
      grid (0.5\rg@grp@width+\rg@padding,%
        0.5\rg@ref@height+\rg@distance+\rg@grp@height+\rg@padding);%
  \fi%
```

Clip the following nodes to graphics width `\rg@grp@width` and clipping height `\rg@clp@height`. This is necessary to get rid of the remaining transparent part of the reflection node, which still consumes much space, even though nothing is visible.

```
  \clip {(-0.5\rg@grp@width,\rg@clp@height-0.5\rg@ref@height)%
    rectangle ++(\rg@grp@width,-\rg@clp@height)};%
```

Draw the graphics node.

```
  \node at (0,\rg@grp@offset) {%
```

Draw graphics using `\includegraphics`.

```
    \includegraphics[%
      width=\rg@width,%
      height=\rg@height,%
      angle=\rg@angle,%
      trim={\rg@triml} {\rg@trimb} {\rg@trimr} {\rg@trimt},%
      clip]{#2}%
  };%
```

Draw the reflection node with previously defined custom fading `fade south` and with the opacity value specified by option `opacity`.

```
  \node at (0,-\rg@ref@offset) [%
    opacity=\rg@opacity,%
    scope fading=fade south,%
    fit fading=false%
  ] {%
```

Flip reflection node content vertically.

```
      \scalebox{1}[-1]{%
```

Draw graphics using `\includegraphics`.

```
        \includegraphics[%
          width=\rg@width,%
          height=\rg@height,%
          angle=\rg@angle,%
          trim={\rg@triml} {\rg@trimb} {\rg@trimr} {\rg@trimt},%
          clip]{#2}%
      }%
    };%
  \end{tikzpicture}%
```

Close group.

```
  \endgroup%
}

\makeatother
```