The hf-tikz package*

Claudio Fiandrino†

July 25, 2014

Abstract

This package provides a way to *highlight* formulas in both documents and presentations thanks to Ti*k*Z. The idea originated in this question on TeX.StackExchange and it is based on the `tikzmark` macro from Andrew Stacey and Peter Grill.

Contents

*This document corresponds to hf-tikz v0.3a, dated 2014/07/25.

†e-mail: `claudio dot fiandrino at gmail dot com`

1 Introduction and requirements

The aim of the package is to provide a simple way to highlight formulas. hf-tikz is not the first package that tries to accomplish this task, but, despite empheq, it provides not only a way to highlight formulas in standard document, but also inside a presentation though Beamer overlay-aware specifications. In addition, with respect to empheq, hf-tikz even allows to highlight just a part of a formula.

The package uses Ti*k*Z and it is based on the `tikzmark` macro from Andrew Stacey and Peter Grill (see this answer and this question): among the numerous versions present on TeX.SX, the one the package implements is taken from this answer. Indeed, as explained later, the concept of *extendible markers* helps a lot in customizing the box dimension.

The packages loaded by hf-tikz are:

- Ti*k*Z and the libraries `shadings` and `decorations.markings` (this library is not always loaded, see subsection 3.6);
- xparse;
- etoolbox.

2 Using the package

2.1 The basic commands

Formulas can be highlighted by inserting a pair of delimiters before and after the part to be highlighted. Two compilation runs are always necessary: the first one to compute the position of the delimiters (also called markers in this manual) and the second one to actually place the box. The starting delimiter should be introduced with the `\tikzmarkin` macro: it may assume a different syntax upon being in `beamer` mode or not as it will be pointed out in subsection 3.1.

`\tikzmarkin`

`\tikzmarkend`

The end delimiter should be introduced by means of the `\tikzmarkend` macro: despite `\tikzmarkin`, this macro keeps the same syntax in `beamer` mode too.

An example of the basic use is:

```
\[x+\tikzmarkin{a}y\tikzmarkend{a}=400\]
```

which produces:

$$x + y = 400$$

Notice that the delimiter labels, also called `marker-ids`, should characterize *uniquely* the part highlighted. Reusing the same name more than once will lead to undesired results. Along this documentation there are examples that illustrates some guidelines to provide names consistenly.

In presence of fractions, sums, integrals and other operators, the standard command is not appropriate. Consider the following example:

```
\[\tikzmarkin{a-1}x+\dfrac{z}{y}=400\tikzmarkend{a-1}\]
```

It leads to:

$$x+\frac{z}{y}=400$$

In this case, the user must specify manually which are the *shift-offsets* that delimits the box:

```
\begin{equation}
\tikzmarkin{right delim frac}(0.1,-0.4)(-0.1,0.5)
x+\dfrac{z}{y}=400
\tikzmarkend{right delim frac}
\end{equation}
```

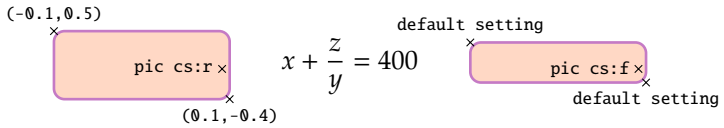
and this fixes the problem:

$$x+\frac{z}{y}=400 \tag{1}$$

The *shift-offsets* should be introduced using the following syntax:

```
\tikzmarkin{marker-id}(below right offset)(above left offset)
```

The following image explains pretty well the difference between the default setting and the *shift-offsets* used in the previous example:



Manual shifts allow to customize the box dimension on the base of user's needs: they should be introduced inside round braces as coordinate points. Coordinates, indeed, provide more degree of freedom from the user's point of view whereas other solutions are more restrictive. Markers, therefore, are *extensible*. Notice that with the aformentioned syntax, it is not possible to use the markers separately, but they should be declared in pair.

From version 0.3, it is also possible to exploit a key-based interface to set the *shift-offsets*; for example, the previous example, could have been done as follows:

```
\begin{equation}
\tikzmarkin[below right offset={0.1,-0.4},above left offset={-0.1,0.5}]
 {right delim frac 2}
x+\dfrac{z}{y}=400
\tikzmarkend{right delim frac 2}
\end{equation}
```

leads to:

$$x+\frac{z}{y}=400 \tag{2}$$

The list of keys available to customize the *shift-offsets* are:

- `left (initial: -0.1/-0.075)`: this key sets the left offset (the second value is active when the `fill` option is passed to the package);

- `right` (`initial: 0.1/0.075`): this key sets the right offset (the second value is active when the `fill` option is passed to the package);
- `above` (`initial: 0.35`): this key sets the above offset;
- `below` (`initial: -0.18`): this key sets the below offset;
- `below right` (`initial: 0.1/0.075,0.35`): this key sets contemporarely the below and the right offsets;
- `above left` (`initial: -0.1/-0.075,0.35`): this key sets contemporarely the above and the left offsets.

All the keys, not only the ones devoted to the *shift-offsets*, should be introduced in the first optional argument only if the `beamer` option is not loaded. In the other case, the first argument of `\tikzmarkin` concerns the overly-specification definition. Furthermore, the keys provided in the optional argument have a local scope. On the contrary, when they are set by means of `\tikzset{}`, they are applied to the whole document. For example:

```
\tikzset{above left offset={-0.1,0.325},below right offset={0.1,-0.4}}
```

2.2 An advanced example

This example shows how to insert an annotation aligned with a sentence: it requires the `calc` library from Ti*k*Z. The colors have been set accordingly to the explanation provided in subsection 3.2.

$$\left.\begin{array}{cc} -2\cdot 2= & -4 \\ -2\cdot 1= & -2 \\ -2\cdot 0= & 0 \end{array}\right\} \text{Product increases by 2 each time.}$$

Annotation

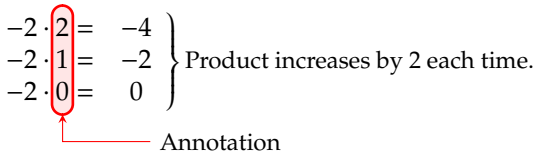
The code is:

```
\begin{equation*}
\left.\begin{array}{cc}
 -2\cdot \tikzmarkin{col}(0.05,-0.3)(-0.05,0.4)2=& -4 \\
 -2\cdot 1=& -2 \\
 -2\cdot 0\tikzmarkend{col}=& 0
\end{array}\right\} \text{\small Product increases by 2 each time.}
\end{equation*}

% To insert the annotation
\begin{tikzpicture}[remember picture,overlay]
 % adjust the shift from "col" to move the position of the annotation
 \coordinate (col-aa) at ($(col)+(1.825,-1.8)$);
 \node[align=left,right] at (col-aa) {\small{Annotation}};
 \path[-stealth,red,draw] (col-aa) -| ($(col)+(0.14,-1.55)$);
\end{tikzpicture}
```

Note that when a formula is highlighted, the `marker-id` can be used to subsequently add elements on the image, i.e. annotations.

From the version 0.3, it exists simpler manner to add annotations that requires the option `markings` to be enabled. Under such an hypothesis, the previous example can be done as follows (the annotation has been put a little bit close to the highlighted area intentionally):



The code:

```
\begin{equation*}
\left.\begin{array}{cc}
 -2\cdot \tikzmarkin[mark at=0.93]{col 1}(0.05,-0.2)(-0.05,0.4)2=& -4 \\
 -2\cdot 1=& -2 \\
 -2\cdot 0\tikzmarkend{col 1}=& 0
\end{array}\right\} \text{\small Product increases by 2 each time.}
\end{equation*}

\begin{tikzpicture}[remember picture,overlay]
 \coordinate (col-aa) at ($(col 1)+(1.3,-1.8)$);
 \node[align=left,right] at (col-aa) {\small{Annotation}};
 \path[-stealth,red,draw,use marker id] (col-aa) -| (0,0);
\end{tikzpicture}
```

Thus, it is sufficient to mark the box delimiting the highlighted area with the option `mark at`. Then, it is possible to access this coordinate by means of `use marker id`. Further details are provided in subsection 3.6.

3 The options

3.1 The `beamer` mode

`beamer` The call:

```
\usepackage[beamer]{hf-tikz}
```

let the package to enter in `beamer` mode and the `\tikzmarkin` macro becomes *overlay-aware*. As a resutl, overlay specifications can be introduced as first argument. For example:

```
\begin{align}
\tikzmarkin<1->{a1}a_i\tikzmarkend{a1} + b_j = 10 \\
\tikzmarkin<3>{c}c_j + d_j +
\tikzmarkin<2>{b}a_i\tikzmarkend{b}
>= 30\tikzmarkend{c}
\end{align}
```

Here it follows a list of examples in which overlay-specifications can be defined:

- a single number: <1>;
- multiple numbers separated by commas and delimited by braces: <{1,2,3}>;
- a single number followed by a dash: <1->.

3.2 Customize colors

customcolors This option allows to customize both the fill and the border color. While using this option, two commands become available:

- \hfsetfillcolor
- \hfsetbordercolor

These commands can be used at any time in the document. For example:

```
\hfsetfillcolor{red!10}
\hfsetbordercolor{red}
\[
\tikzmarkin{z}(0.2,-0.4)(-0.2,0.6)
 \dfrac{100}{x}
\tikzmarkend{z}
\]
```

produces:

$$\frac{100}{x}$$

Then:

```
\hfsetfillcolor{blue!10}
\hfsetbordercolor{blue}
\[\tikzmarkin{z1}x+y=400\tikzmarkend{z1}\]
```

gives:

$$x + y = 400$$

From the version 0.3, it is also possible to customize the fill and the border color by means of the following keys:

- set fill color (initial: fancybrown): this key sets the fill color;
- set border color (initial: fancyviolet): this key sets the border color.

An example:

```
\[
\tikzmarkin[set fill color=green!50!lime!30,
  set border color=green!40!black]{z-a}(0.2,-0.4)(-0.2,0.6)
```

```
 \dfrac{100}{x}
\tikzmarkend{z-a}
\]
```

The result:

$$\boxed{\frac{100}{x}}$$

Notice that:

- the color definition can also be done via `\tikzset`; in this case its application is global in the document

```
\tikzset{set fill color=orange!30,set border color=orange}
```

- global definitions defined via `\tikzset` or `\hfsetfillcolor` and `\hfsetbordercolor` can always be overridden by local ones; that is:

```
\tikzset{set fill color=orange!30,set border color=orange}
\[
\tikzmarkin[set fill color=green!50!lime!30,
  set border color=green!40!black]{label}(0.2,-0.4)(-0.2,0.6)
 \dfrac{100}{x}
\tikzmarkend{z-a}
\]
```

still gives:

$$\boxed{\frac{100}{x}}$$

3.3 Using shadings

shade

The option shade activates the possibility of introducing shaded backgrounds besides any fill color definition currently set up. Available shadings are:

- vertical shading;
- horizontal shading;
- radial shading.

Example with vertical shading

Code:

```
\[
\tikzmarkin[top color=white, bottom color=blue!20]{vshade}
 x+y=400
\tikzmarkend{vshade}
\]
```

Result:

$$x + y = 400$$

Example with horizontal shading

Code:

```
\[
\tikzmarkin[left color=white, right color=blue!20]{hoshade}
 x+y=400
\tikzmarkend{hoshade}
\]
```

Result:

$$x + y = 400$$

Example with radial shading

Code:

```
\[
\tikzmarkin[outer color=white, inner color=blue!20]{rshade}
 x+y=400
\tikzmarkend{rshade}
\]
```

Result:

$$x + y = 400$$

3.4 Avoiding the background color

`nofill` Using the `nofill` option allows very simply to not introduce the background color. When the option is active, you can not change this behaviour inside the document. Another way to remove the background color, is to set the fill color by means of `\hfsetfillcolor` to be of the same color of the page.

3.5 Disable rounded corners

`norndcorners` To disable the rounded corners, it exists two ways actually. The first one, which is a global approach, is the option `norndcorners`: as any of the other package options, it should be provided during the package load.

It exists a second way that disables the rounded corners only locally; this approach needs the `disable rounded corners` key to be set to `true`.

For example:

```
\[
\tikzmarkin[disable rounded corners=true]{mark 1}
 x+y=400
```

```
\tikzmarkend{mark 1}
\]
```

The result:

$$x + y = 400$$

3.6 The `markings` option

`markings` Loading the package with the `markings` option allows to mark positions on the box delimiting the highlighted area. This can be achieved by setting the key `mark at`=<pos> where <pos>=[0,1]; the positions can be later accessed with the key `use marker id`=<id-num> where <id-num> is the progressive identifier of the positions previously marked.

For example:

$$x + y = 400$$

is realized by means of:

```
\tikzset{set fill color=orange!30,set border color=orange}
\[
\tikzmarkin[show markers,mark at=0,]{marker 1}
 x+y=400
\tikzmarkend{marker 1}
\tikz[remember picture,overlay]{
  \draw[use marker id,blue,xscale=-1](0,0)arc(270:90:1.5mm);
}
\]
```

Actually, it is possible to mark more than one point:

$$x + y = 400$$

by referring them in the same order in which they have been marked:

```
\[
\tikzmarkin[show markers,mark at=0,mark at=0.55]{marker 2}
 x+y=400
\tikzmarkend{marker 2}
\tikz[remember picture,overlay]{
  \draw[use marker id=1,blue,xscale=-1](0,0)arc(270:90:1.5mm);
}
\tikz[remember picture,overlay]{
 \draw[use marker id=2,blue](0,0)arc(270:90:1.5mm);
}
\]
```

The markers can become visible when the `show markers` key is activated. By default, they are invisible, but during the working process it may be useful to know their location. In addition, the markers can be customized:

- `marker size` (`initial: 1pt`): this key sets the radius of the marker;
- `marker color` (`initial: blue`): this key sets the color of the marker.

The options necessitates of the `decorations.markings` of Ti*k*Z: this library, however, is not always loaded, but just in case the hf-tikz `markings` option is active.

4 Efficient use of Ti*k*Z styles

Ti*k*Z are very powerful and their use is recommended also with hf-tikz. For example, an intelligent way to proceed if two different highlighting colors have to be used is as follows:

```
\tikzset{offset definition/.style={
    above left offset={-0.1,0.6},
    below right offset={0.1,-0.45},
  },
  h1/.style={
    offset definition,
    set fill color=green!50!lime!60,
    set border color=green!40!black,
  },
  h2/.style={
    offset definition,
    set fill color=blue!20!cyan!60,
    set border color=blue!60!cyan,
  }
}
```

Their use in the document is:

```
\[\tikzmarkin[h1]{st-a}x + \dfrac{y}{z} = 400\tikzmarkend{st-a}\]
\[\tikzmarkin[h2]{st-b}x + \dfrac{y}{z} = 400\tikzmarkend{st-b}\]
```

which gives as result:

$$x + \frac{y}{z} = 400$$

and

$$x + \frac{y}{z} = 400$$

5 Implementation

```
\RequirePackage{tikz}
\usetikzlibrary{shadings}
\RequirePackage{xparse}
\RequirePackage{etoolbox}
```

This warning is arised after the first compilation run to inform that a second run is necessary for the final result. The code has been inspired by this answer on TeX.SX.

```
\AtEndDocument{%
\let\oldpgfsyspdfmark\pgfsyspdfmark
\def\pgfsyspdfmark#1#2#3{%
  \expandafter\let\expandafter\tmp\csname pgf@sys@pdf@mark@pos@#1\endcsname
  \oldpgfsyspdfmark{#1}{#2}{#3}%
  \expandafter\ifx\csname pgf@sys@pdf@mark@pos@#1\endcsname\tmp\else
  \let\oldsavepointas\savepointas
  \def\savepointas##1##2{%
    \immediate\write\@auxout{hf-TikZ Warning: Mark '##1' changed. Rerun to get mark in right position.}%
  }
  \fi
}}
```

5.1 Options definition

This subsection is devoted to define options and default colors.

```
%% Colors

% Pre-defined colors
\definecolor{fancybrown}{RGB}{255,216,197}
\definecolor{fancyviolet}{RGB}{197,122,197}

\newcommand{\fcol}{fancybrown}
\newcommand{\bcol}{fancyviolet}

%% Package option

\newbool{fill}
\booltrue{fill}
\DeclareOption{nofill}{\boolfalse{fill}}

\DeclareOption{customcolors}{
\def\hfsetfillcolor#1{\renewcommand{\fcol}{#1}}
\def\hfsetbordercolor#1{\renewcommand{\bcol}{#1}}
\pgfkeys{/tikz/.cd,
  set fill color/.code={\renewcommand{\fcol}{#1}},
  set border color/.code={\renewcommand{\bcol}{#1}}
 }
}

\newbool{shade}
\boolfalse{shade}
\DeclareOption{shade}{\booltrue{shade}}

\newbool{beamer}
\boolfalse{beamer}
\DeclareOption{beamer}{\booltrue{beamer}}

\newbool{norndcorners}
\boolfalse{norndcorners}
\DeclareOption{norndcorners}{\booltrue{norndcorners}}
```

```
\newbool{markings}
\boolfalse{markings}
\DeclareOption{markings}{\booltrue{markings}}

\ProcessOptions
```

It follows the definition of the keys devoted to disable the rounded corners.

```
\pgfkeys{/tikz/.cd,%
      not use rounded corners/.is choice,%
      not use rounded corners/true/.style={rounded corners=0pt},%
      not use rounded corners/false/.style={rounded corners},%
}%

\tikzset{disable rounded corners/.estyle={%
      not use rounded corners=#1,%
      },%
      disable rounded corners/.default=false,%
}%

```

Offsets keys' definition: for compatibility reasons, the initial values change according to the presence of the `fill` option.

```

\ifbool{fill}{%
 \pgfkeys{/tikz/.cd,%
      left offset/.initial=-0.1,
      right offset/.initial=0.1,
      above offset/.initial=0.35,
      below offset/.initial=-0.18,
 }
}{
 \pgfkeys{/tikz/.cd,%
      left offset/.initial=-0.075,
      right offset/.initial=0.075,
      above offset/.initial=0.35,
      below offset/.initial=-0.18,
 }
}


\pgfkeys{/tikz/.cd,%
      left offset/.get=\leftoff,
      left offset/.store in=\leftoff,
      right offset/.get=\rightoff,
      right offset/.store in=\rightoff,
      above offset/.get=\aboveoff,
      above offset/.store in=\aboveoff,
      below offset/.get=\belowoff,
      below offset/.store in=\belowoff,
```

```
      below right offset/.initial={\rightoff,\belowoff},
      below right offset/.get=\belowrightoff,
      below right offset/.store in=\belowrightoff,
      above left offset/.initial={\leftoff,\aboveoff},
      above left offset/.get=\aboveleftoff,
      above left offset/.store in=\aboveleftoff,
}%
```

Keys and style definition of the markings: they are activated when the `markings` option is present. This is a feature request from Bodo Manthey and the implementation has been inspired by Jake's answer on TeX.SX: thanks to both.

```
\ifbool{markings}{
  \usetikzlibrary{decorations.markings}
  \newif\ifshowmarkers
  \pgfkeys{/tikz/show markers/.is if=showmarkers}
  \pgfkeys{/tikz/show markers=false}

  \pgfkeys{/tikz/.cd,%
      marker color/.initial=blue,
      marker color/.get=\colmarker,
      marker color/.store in=\colmarker,
      marker size/.initial=1pt,
      marker size/.get=\sizemarker,
      marker size/.store in=\sizemarker,
  }

  \tikzset{
   mark at/.style={
     decoration={
        markings,
        mark=
         at position #1
         with
         {
           \coordinate (marker point-\pgfkeysvalueof{/pgf/decoration/mark info/sequence number}) at (0pt,
           \coordinate (marker unit vector-\pgfkeysvalueof{/pgf/decoration/mark info/sequence number}) at
           \coordinate (marker orthogonal unit vector-\pgfkeysvalueof{/pgf/decoration/mark info/sequence
           \ifshowmarkers% conditional to make them appear just when invoked
             \draw[draw=none,fill=\colmarker,radius=\sizemarker] (0,0) circle ;
           \else
             \relax
           \fi
         }
     },
     postaction=decorate
   },
   use marker id/.style={
     shift=(marker point-#1),
     x=(marker unit vector-#1),
     y=(marker orthogonal unit vector-#1)
```

```
    },
    use marker id/.default=1,
  }
}
```

5.2 General settings

This subsection is devoted to illustrate the code used for defining the settings used by the highlighting commands.

```
%% Settings

\ifbool{beamer}{%true
 \newcounter{jumping}
 \resetcounteronoverlays{jumping}

 \def\jump@setbb#1#2#3{%
   \@ifundefined{jump@#1@maxbb}{%
     \expandafter\gdef\csname jump@#1@maxbb\endcsname{#3}%
   }{%
     \csname jump@#1@maxbb\endcsname
     \pgf@xa=\pgf@x
     \pgf@ya=\pgf@y
     #3
     \pgfmathsetlength\pgf@x{max(\pgf@x,\pgf@xa)}%
     \pgfmathsetlength\pgf@y{max(\pgf@y,\pgf@ya)}%
     \expandafter\xdef\csname jump@#1@maxbb\endcsname{\noexpand\pgfpoint{\the\pgf@x}{\the\pgf@y}}%
   }
   \@ifundefined{jump@#1@minbb}{%
     \expandafter\gdef\csname jump@#1@minbb\endcsname{#2}%
   }{%
     \csname jump@#1@minbb\endcsname
     \pgf@xa=\pgf@x
     \pgf@ya=\pgf@y
     #2
     \pgfmathsetlength\pgf@x{min(\pgf@x,\pgf@xa)}%
     \pgfmathsetlength\pgf@y{min(\pgf@y,\pgf@ya)}%
     \expandafter\xdef\csname jump@#1@minbb\endcsname{\noexpand\pgfpoint{\the\pgf@x}{\the\pgf@y}}%
   }
 }

 \tikzset{%
   remember picture with id/.style={%
     remember picture,
     overlay,
     save picture id=#1,
   },
   save picture id/.code={%
     \edef\pgf@temp{#1}%
     \immediate\write\pgfutil@auxout{%
       \noexpand\savepointas{\pgf@temp}{\pgfpictureid}}%
```

```
  },
  if picture id/.code args={#1#2#3}{%
    \@ifundefined{save@pt@#1}{%
      \pgfkeysalso{#3}%
    }{
      \pgfkeysalso{#2}%
    }
  },
  onslide/.code args={<#1>#2}{%
    \only<#1>{\pgfkeysalso{#2}}%
  },
  alt/.code args={<#1>#2#3}{%
    \alt<#1>{\pgfkeysalso{#2}}{\pgfkeysalso{#3}}%
  },
  stop jumping/.style={
    execute at end picture={%
      \stepcounter{jumping}%
      \immediate\write\pgfutil@auxout{%
        \noexpand\jump@setbb{\the\value{jumping}}{\noexpand\pgfpoint{\the\pgf@picminx}{\the\pgf@picminy}
      },
      \csname jump@\the\value{jumping}@maxbb\endcsname
      \path (\the\pgf@x,\the\pgf@y);
      \csname jump@\the\value{jumping}@minbb\endcsname
      \path (\the\pgf@x,\the\pgf@y);
    },
  }
}
}{% false
 \tikzset{%
  remember picture with id/.style={%
    remember picture,
    overlay,
    save picture id=#1,
  },
  save picture id/.code={%
    \edef\pgf@temp{#1}%
    \immediate\write\pgfutil@auxout{%
      \noexpand\savepointas{\pgf@temp}{\pgfpictureid}}%
  },
  if picture id/.code args={#1#2#3}{%
    \@ifundefined{save@pt@#1}{%
      \pgfkeysalso{#3}%
    }{
      \pgfkeysalso{#2}%
    }
  }
}

\def\savepointas#1#2{%
```

```
  \expandafter\gdef\csname save@pt@#1\endcsname{#2}%
}

\def\tmk@labeldef#1,#2\@nil{%
  \def\tmk@label{#1}%
  \def\tmk@def{#2}%
}

\tikzdeclarecoordinatesystem{pic}{%
  \pgfutil@in@,{#1}%
  \ifpgfutil@in@%
    \tmk@labeldef#1\@nil
  \else
    \tmk@labeldef#1,(0pt,0pt)\@nil
  \fi
  \@ifundefined{save@pt@\tmk@label}{%
    \tikz@scan@one@point\pgfutil@firstofone\tmk@def
  }{%
  \pgfsys@getposition{\csname save@pt@\tmk@label\endcsname}\save@orig@pic%
  \pgfsys@getposition{\pgfpictureid}\save@this@pic%
  \pgf@process{\pgfpointorigin\save@this@pic}%
  \pgf@xa=\pgf@x
  \pgf@ya=\pgf@y
  \pgf@process{\pgfpointorigin\save@orig@pic}%
  \advance\pgf@x by -\pgf@xa
  \advance\pgf@y by -\pgf@ya
  }%
}
```

5.3 The highlighting commands

In this subsection the definitions of the highlighing commands are shown when the beamer mode is active and when it is not. Thanks to etoolbox it is possible to perform a check on the active options. Then the commands are declared.

```
\ifbool{norndcorners}{%true-norndcorners
  \ifbool{beamer}{%true-beamer
   \ifbool{fill}{%true-fill
  \ifbool{shade}{%true-shade
   \NewDocumentCommand{\tikzmarkin}{r<> o m D(){\belowrightoff} D(){\aboveleftoff}}{%
   \IfNoValueTF{#2}{%true-val
      \only<#1>{\tikz[remember picture,overlay]
      \draw[line width=1pt,rectangle,fill=\fcol,draw=\bcol]
      (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
      ;}
     }{%false-val
      \only<#1>{\tikz[remember picture,overlay]
      \draw[line width=1pt,rectangle,fill=\fcol,draw=\bcol,#2]
      (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
      ;}}
     }
```

```
}{%false-shade
 \NewDocumentCommand{\tikzmarkin}{r<> o m D(){\belowrightoff} D(){\aboveleftoff}}{%
  \IfNoValueTF{#2}{%true-val
    \only<#1>{\tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,fill=\fcol,draw=\bcol]
    (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
    ;}
   }{%false-val
    \only<#1>{\tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,fill=\fcol,draw=\bcol,#2]
    (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
    ;}}
   }
}
 }{%false-fill
\NewDocumentCommand{\tikzmarkin}{r<> o m D(){\belowrightoff} D(){\aboveleftoff}}{%
  \IfNoValueTF{#2}{%true-val
    \only<#1>{\tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,draw=\bcol]
    (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
    ;}
   }{%false-val
    \only<#1>{\tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,draw=\bcol,#2]
    (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
    ;}}
   }
}
}{%false-beamer
 \ifbool{fill}{%true-fill
\ifbool{shade}{%true-shade
 \NewDocumentCommand{\tikzmarkin}{o m D(){\belowrightoff} D(){\aboveleftoff}}{%
 \IfNoValueTF{#1}{%true-val
    \tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,fill=\fcol,draw=\bcol]
    (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
    ;
 }{%false-val
 \tikz[remember picture,overlay]
 \draw[line width=1pt,rectangle,fill=\fcol,draw=\bcol,#1]
 (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
 ;}}
}{%false-shade
 \NewDocumentCommand{\tikzmarkin}{o m D(){\belowrightoff} D(){\aboveleftoff}}{%
 \IfNoValueTF{#1}{%true-val
    \tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,fill=\fcol,draw=\bcol]
    (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
    ;
 }{%false-val
```

```
   \tikz[remember picture,overlay]
   \draw[line width=1pt,rectangle,fill=\fcol,draw=\bcol,#1]
   (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
   ;}}
  }
   }{%false-fill
    \NewDocumentCommand{\tikzmarkin}{o m D(){\belowrightoff} D(){\aboveleftoff}}{%
   \IfNoValueTF{#1}{%true-val
      \tikz[remember picture,overlay]
      \draw[line width=1pt,rectangle,draw=\bcol]
      (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
      ;
   }{%false-val
   \tikz[remember picture,overlay]
   \draw[line width=1pt,rectangle,draw=\bcol,#1]
   (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
   ;}}
%   \NewDocumentCommand{\tikzmarkin}{m D(){0.075,-0.18} D(){-0.075,0.35}}{%
%   \tikz[remember picture,overlay]
%   \draw[line width=1pt,rectangle,draw=\bcol]
%   (pic cs:#1) ++(#2) rectangle (#3) node [anchor=base] (#1){}
%   ;}
   }
  }
}{%false-norndcorners
\ifbool{beamer}{%true-beamer
 \ifbool{fill}{%true-fill
\ifbool{shade}{%true-shade
 \NewDocumentCommand{\tikzmarkin}{r<> o m D(){\belowrightoff} D(){\aboveleftoff}}{%
 \IfNoValueTF{#2}{%true-val
    \only<#1>{\tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,disable rounded corners,fill=\fcol,draw=\bcol]
    (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
    ;}
   }{%false-val
    \only<#1>{\tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,disable rounded corners,fill=\fcol,draw=\bcol,#2]
    (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
    ;}}
   }
}{%false-shade
 \NewDocumentCommand{\tikzmarkin}{r<> o m D(){\belowrightoff} D(){\aboveleftoff}}{%
   \IfNoValueTF{#2}{%true-val
    \only<#1>{\tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,disable rounded corners,fill=\fcol,draw=\bcol]
    (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
    ;}
   }{%false-val
    \only<#1>{\tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,disable rounded corners,fill=\fcol,draw=\bcol,#2]
```

```
    (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
    ;}}
   }
}
 }{%false-fill
\NewDocumentCommand{\tikzmarkin}{r<> o m D(){\belowrightoff} D(){\aboveleftoff}}{%
 \IfNoValueTF{#2}{%true-val
    \only<#1>{\tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,disable rounded corners,draw=\bcol]
    (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
    ;}
   }{%false-val
    \only<#1>{\tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,disable rounded corners,draw=\bcol,#2]
    (pic cs:#3) ++(#4) rectangle (#5) node [anchor=base] (#3){}
    ;}}
  }
 }
}{%false-beamer
 \ifbool{fill}{%true-fill
\ifbool{shade}{%true-shade
 \NewDocumentCommand{\tikzmarkin}{o m D(){\belowrightoff} D(){\aboveleftoff}}{%
 \IfNoValueTF{#1}{%true-val
    \tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,disable rounded corners,fill=\fcol,draw=\bcol]
    (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
    ;
 }{%false-val
 \tikz[remember picture,overlay]
 \draw[line width=1pt,rectangle,disable rounded corners,fill=\fcol,draw=\bcol,#1]
 (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
 ;}}
}{%false-shade
 \NewDocumentCommand{\tikzmarkin}{o m D(){\belowrightoff} D(){\aboveleftoff}}{%
 \IfNoValueTF{#1}{%true-val
    \tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,disable rounded corners,fill=\fcol,draw=\bcol]
    (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
    ;
 }{%false-val
 \tikz[remember picture,overlay]
 \draw[line width=1pt,rectangle,disable rounded corners,fill=\fcol,draw=\bcol,#1]
 (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
 ;}}
}
 }{%false-fill
\NewDocumentCommand{\tikzmarkin}{o m D(){\belowrightoff} D(){\aboveleftoff}}{%
 \IfNoValueTF{#1}{%true-val
    \tikz[remember picture,overlay]
    \draw[line width=1pt,rectangle,disable rounded corners,draw=\bcol]
```

```
    (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
    ;
 }{%false-val
 \tikz[remember picture,overlay]
 \draw[line width=1pt,rectangle,disable rounded corners,draw=\bcol,#1]
 (pic cs:#2) ++(#3) rectangle (#4) node [anchor=base] (#2){}
 ;}}
 }
}
}

\newcommand\tikzmarkend[2][]{%
\tikz[remember picture with id=#2] #1;}
```