

The `htikz` package*

Claudio Fiandrino[†]

April 28, 2012

Abstract

This package contains some useful commands to *highlight* formulas in documents and presentations thanks to TikZ. The idea comes out from [this question](#) and actually the package is just an adaptation of the code provided by [Andrew Stacey](#) in [this answer](#).

Contents

1	The usage	1
2	The options	2
3	Implementation	3
3.1	Options definition	3
3.2	General settings	4
3.3	The highlighting commands	6

1 The usage

`\tikzmarkin` The way in which it is possible to highlight formulas is thanks to the insertion of delimiters before and after the part to be highlighted. To start delimiting the formula you should use the macro `\tikzmarkin` which can behave differently upon being in beamer mode or not. To end delimiting the formula to be highlighted you should use the macro `\tikzmarkend`: despite `\tikzmarkin`, this macro keep the same behavior also in beamer mode. An example of the basic usage is:

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

*This document corresponds to `htikz` v0.1, dated 2012/04/25.

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

which produces:

$$x + y = 400$$

When you wonder to highlight fractions, integrals or sums, the basic commands are not suitable. You should use the *extended* ones:

- `\exttikzmarkin`
- `\exttikzmarkend`

An example of this usage is:

```
\[\exttikzmarkin{r}\dfrac{100}{x}\exttikzmarkend{r}\]
```

2 The options

beamer When you call the package:

```
\usepackage[beamer]{htikz}
```

you enter in beamer mode and the `\tikzmarkin` and `\exttikzmarkend` macro allows you to decide when to insert the highlighting. 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}
```

nofill Using the option `nofill` allows you to simply not see the color in background. When the option is active, you can not change this behaviour inside the document.

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

- `\setfillcolor`
- `\setbordercolor`

They can be use in whatever part of the document allowing a high customization of colors. For example:

```
\setfillcolor{red!10}
\setbordercolor{red}
\[\exttikzmarkin{c}\dfrac{100}{x}\exttikzmarkend{c}\]
```

produces:

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

Then:

```
\setfillcolor{blue!10}  
\setbordercolor{blue}  
\[\tikzmarkin{x}x+y=400\tikzmarkend{x}\]
```

produces:

$$x + y = 400$$

Notice that, once `customcolors` is active, you need to specify colors at least in the preamble.

3 Implementation

```
1 \NeedsTeXFormat{LaTeX2e}  
2 \ProvidesPackage{htikz}[2012/04/25 v0.1 A simple way to highlight formulas parts.]  
3 \RequirePackage{tikz}  
4 \RequirePackage{etoolbox}
```

3.1 Options definition

In this subsection we show the definitions of pre-defined colors and options.

```
5 %% Colors  
6  
7 % Pre-defined colors  
8 \definecolor{lightbrown}{RGB}{255,218,195}  
9 \definecolor{violet}{RGB}{197,122,195}  
10  
11 \newcommand{\fcol}{lightbrown}  
12 \newcommand{\bcol}{violet}  
13  
14 %% Package option  
15  
16 % Decide whether to fill or not the highlighting  
17 \newbool{fill}  
18 \booltrue{fill}  
19 \DeclareOption{nofill}{\boolfalse{fill}}  
20  
21 % Decide whether to change pre-defined colors  
22 \DeclareOption{customcolors}{  
23 \def\setfillcolor#1{\def\@fillcolor{#1}}  
24 \def\setbordercolor#1{\def\@bordercolor{#1}}  
25  
26 \renewcommand{\fcol}{\@fillcolor}  
27 \renewcommand{\bcol}{\@bordercolor}  
28 }  
29  
30 % Usage inside beamer class  
31 \newbool{beamer}
```

```

32 \boolfalse{beamer}
33 \DeclareOption{beamer}{\booltrue{beamer}}
34
35 \ProcessOptions

```

3.2 General settings

In this subsection we show the general settings that allow the highlighting.

```

36 %% Settings
37
38 \ifbool{beamer}{%true
39   \newcounter{jumping}
40   \resetcounteronoverlays{jumping}
41
42   \def\jump@setbb#1#2#3{%
43     \@ifundefined{jump@#1@maxbb}{%
44       \expandafter\gdef\csname jump@#1@maxbb\endcsname{#3}%
45     }{%
46       \csname jump@#1@maxbb\endcsname
47       \pgf@xa=\pgf@x
48       \pgf@ya=\pgf@y
49       #3
50       \pgfmathsetlength\pgf@x{max(\pgf@x,\pgf@xa)}%
51       \pgfmathsetlength\pgf@y{max(\pgf@y,\pgf@ya)}%
52       \expandafter\xdef\csname jump@#1@maxbb\endcsname{\noexpand\pgfpoint{\the\pgf@x}{\the\pgf@y}}
53     }
54     \@ifundefined{jump@#1@minbb}{%
55       \expandafter\gdef\csname jump@#1@minbb\endcsname{#2}%
56     }{%
57       \csname jump@#1@minbb\endcsname
58       \pgf@xa=\pgf@x
59       \pgf@ya=\pgf@y
60       #2
61       \pgfmathsetlength\pgf@x{min(\pgf@x,\pgf@xa)}%
62       \pgfmathsetlength\pgf@y{min(\pgf@y,\pgf@ya)}%
63       \expandafter\xdef\csname jump@#1@minbb\endcsname{\noexpand\pgfpoint{\the\pgf@x}{\the\pgf@y}}
64     }
65   }
66
67   \tikzset{%
68     remember picture with id/.style={%
69       remember picture,
70       overlay,
71       draw=\bcol,
72       save picture id=#1,
73     },
74     save picture id/.code={%
75       \edef\pgf@temp{#1}%
76       \immediate\write\pgfutil@auxout{%

```

```

77         \noexpand\savepointas{\pgf@temp}{\pgfpictureid}}%
78     },
79     if picture id/.code args={#1#2#3}{%
80         \@ifundefined{save@pt@#1}{%
81             \pgfkeysalso{#3}%
82         }{
83             \pgfkeysalso{#2}%
84         }
85     },
86     onslide/.code args={<#1>#2}{%
87         \only<#1>{\pgfkeysalso{#2}}%
88     },
89     alt/.code args={<#1>#2#3}{%
90         \alt<#1>{\pgfkeysalso{#2}}{\pgfkeysalso{#3}}%
91     },
92     stop jumping/.style={
93         execute at end picture={%
94             \stepcounter{jumping}%
95             \immediate\write\pgfutil@auxout{%
96                 \noexpand\jump@setbb{\the\value{jumping}}{\noexpand\pgfpoint{\the\pgfpicminx}{\the\pgfpicmaxx}}%
97             },
98             \csname jump@\the\value{jumping}@maxbb\endcsname
99             \path (\the\pgf@x,\the\pgf@y);
100             \csname jump@\the\value{jumping}@minbb\endcsname
101             \path (\the\pgf@x,\the\pgf@y);
102         },
103     }
104 }
105 }{% false
106     \tikzset{%
107         remember picture with id/.style={%
108             remember picture,
109             overlay,
110             draw=\bcol,
111             save picture id=#1,
112         },
113         save picture id/.code={%
114             \edef\pgf@temp{#1}%
115             \immediate\write\pgfutil@auxout{%
116                 \noexpand\savepointas{\pgf@temp}{\pgfpictureid}}%
117         },
118         if picture id/.code args={#1#2#3}{%
119             \@ifundefined{save@pt@#1}{%
120                 \pgfkeysalso{#3}%
121             }{
122                 \pgfkeysalso{#2}%
123             }
124         }
125     }
126 }

```

```

127
128 \def\savepointas#1#2{%
129   \expandafter\gdef\csname save@pt@#1\endcsname{#2}%
130 }
131
132 \def\tmk@labeldef#1,#2\@nil{%
133   \def\tmk@label{#1}%
134   \def\tmk@def{#2}%
135 }
136
137 \tikzdeclarecoordinatesystem{pic}{%
138   \pgfutil@in@,{#1}%
139   \ifpgfutil@in@%
140     \tmk@labeldef#1\@nil
141   \else
142     \tmk@labeldef#1,\pgfpointorigin\@nil
143   \fi
144   \@ifundefined{save@pt@\tmk@label}{%
145     \tikz@scan@one@point\pgfutil@firstofone\tmk@def
146   }{%
147     \pgfsys@getposition{\csname save@pt@\tmk@label\endcsname}\save@orig@pic%
148     \pgfsys@getposition{\pgfpictureid}\save@this@pic%
149     \pgf@process{\pgfpointorigin\save@this@pic}%
150     \pgf@xa=\pgf@x
151     \pgf@ya=\pgf@y
152     \pgf@process{\pgfpointorigin\save@orig@pic}%
153     \advance\pgf@x by -\pgf@xa
154     \advance\pgf@y by -\pgf@ya
155   }%
156 }

```

3.3 The highlighting commands

In this subsection we show the definition of the highlighting commands in `beamer` mode and not. When the `nofill` option is active, there is no definition for the `fill` color.

```

157 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
158 % The highlighting commands
159
160 \ifbool{beamer}{%true
161   \ifbool{fill}{%true
162     \newcommand<>{\tikzmarkin}[1]{%
163       \only#2{\tikz[remember picture with id=#1]
164         \draw[line width=1pt,rectangle,rounded corners,fill=\fcol]
165         (pic cs:#1) ++(0.1,-0.18) rectangle (-0.1,0.35)
166       ;}}
167     \newcommand<>{\exttikzmarkin}[1]{%
168       \only#2{\tikz[remember picture with id=#1]
169         \draw[line width=1pt,rectangle,rounded corners,fill=\fcol]

```

```

170     (pic cs:#1) ++(0.1,-0.5) rectangle (-0.1,0.65)
171     ;}}
172   }{%false
173     \newcommand<>{\tikzmarkin}[1]{%
174       \only#2{\tikz[remember picture with id=#1]
175         \draw[line width=1pt,rectangle,rounded corners]
176         (pic cs:#1) ++(0.075,-0.18) rectangle (-0.075,0.35)
177         ;}}
178     \newcommand<>{\exttikzmarkin}[1]{%
179       \only#2{\tikz[remember picture with id=#1]
180         \draw[line width=1pt,rectangle,rounded corners]
181         (pic cs:#1) ++(0.1,-0.5) rectangle (-0.1,0.65)
182         ;}}
183   }
184 }{%false
185   \ifbool{fill}{%true
186     \newcommand{\tikzmarkin}[1]{%
187       \tikz[remember picture with id=#1]
188       \draw[line width=1pt,rectangle,rounded corners,fill=\fcol]
189       (pic cs:#1) ++(0.1,-0.18) rectangle (-0.1,0.35)
190       ;}
191     \newcommand{\exttikzmarkin}[1]{%
192       \tikz[remember picture with id=#1]
193       \draw[line width=1pt,rectangle,rounded corners,fill=\fcol]
194       (pic cs:#1) ++(0.1,-0.5) rectangle (-0.1,0.65)
195       ;}
196   }{%false
197     \newcommand{\tikzmarkin}[1]{%
198       \tikz[remember picture with id=#1]
199       \draw[line width=1pt,rectangle,rounded corners]
200       (pic cs:#1) ++(0.075,-0.18) rectangle (-0.075,0.35)
201       ;}
202     \newcommand{\exttikzmarkin}[1]{%
203       \tikz[remember picture with id=#1]
204       \draw[line width=1pt,rectangle,rounded corners]
205       (pic cs:#1) ++(0.1,-0.5) rectangle (-0.1,0.65)
206       ;}
207   }
208
209 }
210
211 \newcommand\tikzmarkend[2] []{%
212 \tikz[remember picture with id=#2] #1;}
213
214 \newcommand\exttikzmarkend[2] []{%
215 \tikz[remember picture with id=#2] #1;}

```