The onimage package*

TeX.SX

April 15, 2022

1 Drawing on top of an image

The question "Drawing on an image with TikZ" asks for a method to draw on top of an image file (included by `\includegraphics`) with TikZ. The basic solution to this problem is to include the image in a node inside the `tikzpicture` environment and add TikZ `\draw` commands as one normally would. To make this process easy it is useful to set the coordinate system relative to the size of the picture, so that the draw graphics are scaled whenever the image is scaled.

`tikzonimage`

The `tikzonimage` environment is used to include an image file and then use TikZ to draw on top of it. It starts a new `tikzpicture` environment and sets the coordinate system so that the origin is at the lower left corner of the image. By default it also scales the coordinate system so that the point $(1, 1)$ is at the upper right corner of the image.

`\begin{\tikzonimage}[`⟨*image options*⟩`]{`⟨*image file*⟩`}[`⟨*TikZ options*⟩`]`

The contents of the first optional argument are are passed to `\includegraphics`, the second optional argument works in exactly the same way as the optional argument of the `tikzpicture` environment. For example, in order to draw a small red circle in the middle of `some_image.jpg`, one could write

```
\begin{tikzonimage}[width=0.3\textwidth]{some_image.jpg}[color=red]
    \fill (0.5,0.5) circle [radius=2pt];
\end{tikzonimage}
```

Note that the radius has to be given with an absolute unit, so that the circle does not get squeezed into an ellipse if `some_image.jpg` is not square.

`tsx/scale cs`

The coordinate system scaling behaviour can be modified with the `tsx/scale cs` option. Its default value is `both`, resulting in the behaviour described above. If the option is set to `x`, then the coordinate system as set so that $(1, 0)$ is at the lower right corner of the image and the rectangle with vertices $(0, 0)$, $(0, 1)$, $(1, 1)$ and $(1, 0)$ is a square. This is demonstrated in Figure **??**. Setting `tsx/scale cs=y` works analogously. If no scaling is desired at all, it can be disabled by setting the option to `none`.

`tsx/show help lines`

For easier placement of graphics on top of the image, it is often desirable to display a grid of lines on top of it during the development process. This can be achieved with the `tsx/show help lines` option, which causes the package to display a grid with coordinate labels on top of the image. An example is shown in Figure **??**. The option takes an argument to specify how many lines are shown

*This document corresponds to onimage v0.3, dated 2011/07/05.

Figure 1: The `tsx/scale cs=x` option.

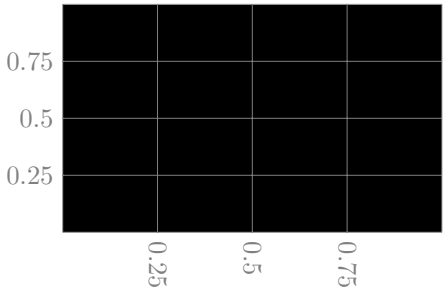

Figure 2: The `tsx/show help lines` option.

(technically this is not quite true: the number displayed of lines is one higher; for example, Figure **??** is created with `tsx/show help lines=4`). The default value is 10. If the argument is not an integer, then it is interpreted as a dimension specifying the distance between the lines.

`tikzonnode` Sometimes one would like to include several images into a `tikzpicture` and still have the rescaled coordinate system for each of them. For this reason this package also provides an environment to change the coordinate system so that it matches an arbitrary node.

`\begin{\tikzonnode}{`⟨*node*⟩`}[`⟨*TikZ options*⟩`]`

The `tikzonnode` environment acts like a `scope` so that, for example, it limits clipping. To draw a red circle in the middle of a node with some text, one could write

```
\begin{tikzpicture}
    \node (A) [text width=2cm] {abc abc abc abc abc abc abc abc abc};
    \begin{tikzonnode}{A}[color=red]
        \fill (0.5,0.5) circle [radius=2pt];
    \end{tikzonnode}
\end{tikzpicture}
```

The environment respects the `tsx/scale cs` and `tsx/show help lines` options.

2 Implementation

```
\RequirePackage{tikz}
\RequirePackage{xstring}

```

These options control the scaling of the coordinate system in the `tikzonnode` environment.

```
\newif\iftsx@scalecs@x
\newif\iftsx@scalecs@y
\tikzset{
    tsx/scale cs/.is choice,
    tsx/scale cs/x/.code={\tsx@scalecs@xtrue\tsx@scalecs@yfalse},
    tsx/scale cs/y/.code={\tsx@scalecs@xfalse\tsx@scalecs@ytrue},
    tsx/scale cs/both/.code={\tsx@scalecs@xtrue\tsx@scalecs@ytrue},
    tsx/scale cs/none/.code={\tsx@scalecs@xfalse\tsx@scalecs@yfalse},
    tsx/scale cs=both
}

```

The next option controls the display of the help lines. By default they are disabled.

```
\tikzset{
    tsx/show help lines/.initial=0,
    tsx/show help lines/.default=10
}
```

`tikzonnode` The implementation is pretty straightforward. `tikzonnode` simply creates a `scope` with the options for the coordinate system shift and passes the optional argument.

```
\def\tikzonnode#1{%
    \pgfutil@ifnextchar[{\tikzonnode@opt#1}{\tikzonnode@opt#1[]}%
}
\def\tikzonnode@opt#1[#2]{%
    \pgfpointanchor{#1}{south west}%
    \pgfgetlastxy\tse@tikz@shift@x\tse@tikz@shift@y
    \begin{scope}[
            shift={(\tse@tikz@shift@x,\tse@tikz@shift@y)},
            #2]%
```

Note that PGF stores the `x` and `y` vector in `\pgf@xx`, `\pgf@xy`, etc. For simplicity we set some of these values directly.

```
        \iftsx@scalecs@x
            \tikzset{x={(#1.south east)}}
            \iftsx@scalecs@y\else
                \pgf@yx=\pgf@xy
                \pgf@yy=\pgf@xx
            \fi
        \fi
        \iftsx@scalecs@y
            \tikzset{y={(#1.north west)}}
            \iftsx@scalecs@x\else
                \pgf@xx=\pgf@yy
                \pgf@xy=\pgf@yx
            \fi
        \fi
```

Draw the help lines, if requested.

```
        \begingroup
            \pgfkeys{/pgf/number format/.cd,fixed,precision=2}
```

```
            \tikzset{tsx/show help lines/.get=\tsx@helplines}
            \IfInteger\tsx@helplines{
```

If `show help lines` is set to an integer, just draw that many evenly spaced lines.

```
                \ifnum\tsx@helplines>1
                    \pgfmathsetmacro\tsx@stepsize{1/\tsx@helplines}
                    \draw[help lines,xstep=\tsx@stepsize,ystep=\tsx@stepsize] (0,0) grid (1,1)
                    \pgfmathsetmacro\tsx@numlines{\tsx@helplines - 1}
                    \foreach \i in {1,...,\tsx@numlines} {
                        \pgfmathsetmacro\tsx@step{\i*\tsx@stepsize}
                        \node [help lines, anchor=west,rotate=-90] at (\tsx@step,0) {\pgfmathp
                        \node [help lines, anchor=east] at (0,\tsx@step) {\pgfmathprintnumber{
                    }
                \fi
            }{
```

If it is a dimension, draw lines that much apart.

```
                \let\tsx@stepsize\tsx@helplines

                \pgfpointanchor{#1}{south west}
                \pgfgetlastxy\tsx@sw@x\tsx@sw@y
                \pgfpointanchor{#1}{north east}
                \pgfgetlastxy\tsx@ne@x\tsx@ne@y
                \pgfmathsetmacro\tsx@width{\tsx@ne@x-\tsx@sw@x}
                \pgfmathsetmacro\tsx@height{\tsx@ne@y-\tsx@sw@y}

                \IfDecimal{\tsx@stepsize}{
                    \let\tsx@stepsize@multx=\pgf@xx
                    \let\tsx@stepsize@multy=\pgf@yy
                }{
                    \def\tsx@stepsize@multx{1}
                    \def\tsx@stepsize@multy{1}
                }
                \pgfmathsetmacro\tsx@numlinesx{floor(\tsx@width/(\tsx@stepsize*\tsx@stepsize@m
                \pgfmathsetmacro\tsx@numlinesy{floor(\tsx@height/(\tsx@stepsize*\tsx@stepsize@

                \ifdim\tsx@numlinesx pt>0pt
                \ifdim\tsx@numlinesy pt>0pt
                    \draw[help lines,xstep=\tsx@stepsize,ystep=\tsx@stepsize] (0,0) grid (\tsx
                    \foreach \x in {1,...,\tsx@numlinesx} {
                        \pgfmathsetmacro\tsx@step{\x*\tsx@stepsize*\tsx@stepsize@multx}
                        \node [help lines, anchor=west,rotate=-90] at (\tsx@step pt,0) {\x};
                    }
                \foreach \y in {1,...,\tsx@numlinesy} {
                    \pgfmathsetmacro\tsx@step{\y*\tsx@stepsize*\tsx@stepsize@multy}
                    \node [help lines, anchor=east] at (0,\tsx@step pt) {\y};
                }
                \fi
                \fi
                }
        \endgroup
}
\def\endtikzonnode{%
    \end{scope}%
}
```

`tikzonimage` To draw on a picture, we simply include it in a node and use `tikzonnode` to set the coordinate system.

```
\def\tikzonimage{%
    \pgfutil@ifnextchar[{\tikzonimage@opt}{\tikzonimage@opt[]}%
}
\def\tikzonimage@opt[#1]#2{%
    \begin{tikzpicture}
        \node[inner sep=0] (image) {\includegraphics[#1]{#2}};
        \begin{tikzonnode}{image}%
}
\def\endtikzonimage{%
        \end{tikzonnode}
    \end{tikzpicture}%
}
```