The \texttt{tikzmark} package

Andrew Stacey

\texttt{loospaces@mathforge.org}

v1.8 from 2019/10/04

1 Introduction

The \texttt{tikzmark} macro burst onto the scene in a blaze of glory on \TeX-
SX. Since then, it has proved embarrassingly (to its original author) popular. The idea
behind it is extremely simple: that the machinery underneath TikZ provides a
way to “mark” a point on a page for further use. This functionality is already
provided by several other packages. The point of this one is that as TikZ can
provide this feature, if already loading TikZ then it makes sense to use the TikZ
version than another version. Moreover, if the goal is to use these marks with
some TikZ code then this version is already set up for that purpose (not that it
would be exactly difficult to add this to any of the other implementations).

2 Use

Using the \texttt{tikzmark} is extremely simple. You need to load the \texttt{tikz}
package and then load \texttt{tikzmark} as a \texttt{tikzlibrary}. Thus in your preamble you should have
something like:

\begin{verbatim}
usepackage{tikz}
usetikzlibrary{tikzmark}
\end{verbatim}

In your document, you can now type \texttt{tikzmark\{<name>\}} at a point that you
want to remember. This will save a mark with name \texttt{<name>} for use later (or
earlier). To use it in a \texttt{tikz} or \texttt{tikzpicture}, simply use the \texttt{pic} coordinate
system:

\begin{verbatim}
\tikz[remember picture] \draw[overlay] (0,0) -- (pic cs:<name>);
\end{verbatim}

There are two important points to note:

1. The enveloping \texttt{tikz} or \texttt{tikzpicture} must have the key \texttt{remember picture}
   set.

   This is because of how TikZ coordinates work. The coordinates inside a
   TikZ picture are relative to its origin, so that origin can move around on
   the page and not affect the internals of the picture. To use a point outside

1
the picture, therefore, the current picture not only has to know where that point is on the page it also has to know where it itself is on the page. Hence the *remember picture* key must be set.

2. The drawing command must have the *overlay* key set (or be in a scope or picture where it is set).

This is to keep the bounding box of the current picture under control. Otherwise, it would grow to encompass the remembered point as well as the current picture. (This isn’t necessary if the remembered point is inside the current picture.)

3 History

I wrote the original *tikzmark* macro in 2009 for use in lecture slides prepared with the *beamer* package. Its original definition was:

\begin{verbatim}
\newcommand*{\tikzmark}{\tikz[overlay,remember picture] \node (#1) {};}\end{verbatim}

Its first use was in the (inelegant) code:

\begin{verbatim}
\begin{frame}
\frametitle{Structure of Continuous Functions}
\begin{tikzpicture}[overlay, remember picture]
\useasboundingbox (0,0);
\draw<2-|trans: 0|handout: 0>[red,->] (bsp) .. controls +(-1,-1) and
\quad ($(cnvs.north)+(1,1)$) .. ($($cnvs.north)+(0,1)$) .. controls
\quad ($(cnvs.north)+(1,1)$) and +(-1,0) .. (cnvs.north);
\draw<3-|trans: 0|handout: 0>[green!50!black,->] (cplt) .. controls
\quad +(-1,-1) and +(-1,0) .. (mcplt.north);
\draw<4-|trans: 0|handout: 0>[blue,->] (norm) .. controls +(-1,-.5) and
\quad ($(nvs.north)+(0,1.5)$) .. ($($nvs.north)+(0,1.5)$) .. controls
\quad ($(nvs.north)+(-1.5,1.5)$) and +(-1.5,0) .. (nvs.north);
\draw<5-|trans: 0|handout: 0>[purple,->] (vector) .. controls +(-1,-1) and
\quad ($(vsp.north)+(2,2)$) .. ($($vsp.north)+(0,2)$) .. controls
\quad ($(vsp.north)+(2,2)$) and +(-2,0) .. (vsp.north);
\end{tikzpicture}
\begin{theorem}
\centering
\big(C([0,1],\mathbb{R}),d_\infty\big) is a \alert{Banach} space\end{theorem}
\pause
\bigskip
\begin{itemize}
\item
\end{itemize}
\end{verbatim}
Structure of Continuous Functions

Theorem

\[(C([0, 1], \mathbb{R}), d_{\infty})\]

is a Banach space

Complete normed vector space.

Cauchy sequences converge.

Metric from a norm.

Functions behave like vectors.

Figure 1: First use of tikzmark

This produced, on the final slide, Figure 1
Its first appearance on TeX-SX was in an answer to a question about how to put overlapping braces on a mathematical text. This was in July 2010. The opening statement of the answer was not overly encouraging: “This may not be the best solution...”. And for a macro that would go on to become quite ubiquitous, its initial appearance only garnered it 2 votes.

However, it started out in life as a useful macro for me and as such I found more uses for it in my own code and thus more opportunity for using it to answer questions on TeX-SX. The one that seems to have been where it got noticed came in August 2010, again about putting braces in text but in a more complicated fashion. From this answer, it got picked up, picked over, and picked apart. A common use was in highlighting or adding marks to text.

Gradually, as it got used, it developed. A major revision dates from an answer given in March 2012 where the question was actually about \tikzmark. This version added two important features: a TikZ coordinate system for referencing saved marks directly and the ability to refer to marks earlier in the document than they are defined (the mechanism for remembering points uses the aux file anyway so this was more about exposing the information earlier than anything complicated). Then in October 2012 there was a question where it would have been useful to remember which page the mark was on and a question where for some reason using the \tikz macro didn’t work so the \pgfmark macro was introduced.

By this point, the \tikzmark command had morphed considerably from its original definition. Experience has shown that on the TeX-SX site it has continued to be used in its original form as well as its current form. I’ve therefore taken the decision to reintroduce a form of the original command, now called \tikzmarknode. It goes beyond the original version in that it uses some \mathchoice trickery (inspired by this answer from Heiko Oberdiek) to hopefully correctly choose the correct math style.

The original reason for not using nodes inside \tikzmark was to be able to use the information from a \tikzmark before the point where it was defined (via information saved into the aux file). Thanks to a question on TeX-SX about saving node information, I’ve developed code that solves that issue with nodes. As it fits in the general concept of this package, I’ve added that code to the \tikzmark package.

4 Usage

This package defines the following commands and usable stuff.

1. \tikzmark\[\langle\text{drawing command}\rangle\{\langle\text{name}\rangle\}

The mandatory argument is the name of the mark to be used to refer back to this point later.

The \tikzmark command can take an optional parameter which is some drawing command that can be put in a \tikz \dots; command. This drawing command can be used to place a node or something similar at the marked point, or to set some \tikzset keys. Sometimes this can be useful. Note,
though, that if this is used to define an offset coordinate then this will only be available in the document after the \tikzmark command, even on later runs.

If the beamer class is loaded then this command is made overlay-aware.

2. \tikzmark{⟨name⟩}{⟨coordinate⟩}

v1.2 of the tikzmark package introduced a new variant of \tikzmark which works inside a tikzpicture. One feature of \tikzmark which isn’t part of TikZ’s normal coordinate remembering system is the ability to use a \tikzmark coordinate before it is defined (due to the use of the aux file). This is potentially useful to have inside a tikzpicture and so it is now possible to use \tikzmark inside a tikzpicture. The syntax is slightly different as we need to specify the coordinates of a point to remember.

This was inspired by the question Refer to a node in tikz that will be defined “in the future” (two passes)? on TeX-SX.

3. \pgfmark{⟨name⟩}

This is a more basic form of the \tikzmark which doesn’t use any of the \tikz overhead. One advantage of this command is that it doesn’t create an hbox. It does, however, insert a whatsit into the stream so it will, for example, stop two vertical spaces either side of it being merged. This can’t be avoided.

If the beamer class is loaded then this command is made overlay-aware.

4. \iftikzmark{⟨name⟩}{⟨true code⟩}{⟨false code⟩}

This is a conditional to test if a particular mark is available. It executes true code if it is and false code if not.

5. \iftikzmarkexists{⟨name⟩}

This is a conditional to test if a particular mark is available which works with the lower level \TeX \else and \fi.

6. \iftikzmarkoncurrentpage{⟨name⟩}

This is a conditional to test if a particular mark is on the current page; it works with the lower level \TeX \else and \fi.

7. \iftikzmarkonpage{⟨name⟩}{⟨page⟩}

This is a conditional to test if a particular mark is on a given page; it works with the lower level \TeX \else and \fi.

8. \tikzmarknode[⟨options⟩]{⟨name⟩}{⟨contents⟩}

This is a reincarnation of the original \tikzmark command which places its contents inside a \tikz node. It also defines a tikzmark with the same name. Using a sneaky trick with \mathchoice, it works inside a math environment.
The spacing either side might not be quite right as although it detects the math style it doesn’t go beyond that. The options are passed to the node.

Two styles are attempted, one on the surrounding picture and one on the node, which are:

- every tikzmarknode picture
- every tikzmarknode

To refer to the node, use usual TikZ coordinates. To refer to the underlying tikzmark, use the special tikzmark coordinates (see below).

9. (pic cs:<name>) or (pic cs:<name>,<coordinate>)

This is the method for referring to a position remembered by \tikzmark (or \pgfmark) as a coordinate in a tikzpicture environment (or \tikz command). If the extra coordinate is specified then this is used in case the mark name has not yet been defined (this can be useful for defining code that does something sensible on the first run).

10. /tikz/save picture id=<name>

This is the TikZ key that is used by \tikzmark to actually save the connection between the name and the picture coordinate. It can be used on an arbitrary picture to save its origin as a tikzmark.

11. /tikz/if picture id=#1#2#3

This is a key equivalent of the \iftikzmark command.

12. /tikz/if tikzmark on current page=#1#2#3

This is a key equivalent of the \iftikzmarkoncurrentpage command. If true, the keys in #2 are executed, otherwise the keys in #3.

13. /tikz/if tikzmark on page=#1#2#3#4

This is a key equivalent of the \iftikzmarkonpage command.

14. /tikz/next page, /tikz/next page vector

It is possible to refer to a mark on a different page to the current page. When this is done, the mark is offset by a vector stored in the key /tikz/next page vector. The key /tikz/next page can be used to set this to certain standard vectors by specifying where the “next page” is considered as lying corresponding to the current page. Possible values are (by default) above, below, left, right, and ignore. (The last one sets the vector to the zero vector.)

Previous versions of tikzmark tried to make this work correctly with the mark being on, say, 5 pages further on but this got too fiddly so this version just pretends that the mark is on the next or previous page and points to it as appropriate.
15. \tikz/tikzmark prefix=<prefix> and \tikz/tikzmark suffix=<suffix>

These keys allow for the automatic addition of a prefix and/or suffix to each \tikzmark name. The prefix and suffix are added both at time of definition and of use, so providing one is in the same scope there is no difference in the user level when using prefixes and suffixes. What it can be useful for is to make the \tikzmark names unique. In particular, if the beamer class is loaded then an automatic suffix is added corresponding to the overlay. This means that if a slide consists of several overlays with \tikzmarks on them, and the positions of the \tikzmarks move then the resulting pictures should look right. Without the automatic suffix, only the final positions of the marks would be used throughout.

This was inspired by the question [using tikzmark subnode with overlays beamer](#) on TeX-SX.

16. \subnode[options]{name}{content}

This produces a pseudo-node named name around the content. The design purpose of this is to create a “subnode” inside a TikZ node. As far as TikZ is concerned, the contents of a node is just a box. It therefore does not know anything about it beyond its external size and so cannot easily determine the coordinates of pieces inside. The \subnode command boxes its contents and saves the position of that box and its dimensions. This information is stored in the same way that PGF stores the necessary information about a node. It is therefore possible to use ordinary node syntax (within a \tikzpicture) to access this information. Thus after \node {a \subnode{a}{sub} node}; it is possible to use a as a node. The options are passed to the node construction mechanism, but note that the only sensible options are those that affect the size and shape of the node: drawing options are ignored (except in so far as they affect the size – as an example, line width affects the node size).

There are two important points to make about this. The first is that, as with all the \tikzmark macros, the information is always one compilation old. The second is that the pseudo-node is purely about coordinates: the path information is not used and the contents are not moved. This is partly for reasons of implementation: the pseudo-node is constructed when TikZ is not in “picture mode”. But also interleaving the background path of the pseudo-node and any containing node would be problematic and so is best left to the user.

The simplest way to turn a pseudo-node into a more normal node is to use the \fit library. Using the above example, \node[fit=(a),draw,inner sep=0pt] {}; would draw a rectangle around the word sub of exactly the same size as would appear had a normal node been created.

Using a sneaky trick with \mathchoice, \subnode works inside a math environment. The spacing either side might not be quite right as although it detects the math style it doesn’t go beyond that.
17. **Node saving**

The node saving system takes the information stored about a node and saves it for later use. That later use can be in the same document, in which case it should be saved just to the memory of the current TeX process, or it can be used earlier in the same document or another document altogether (in particular, if the nodes are defined in a `tikzpicture` that has been externalised, this can be used to import the node information into the main file) in which cases the node data is saved to a file.

When working with files, nodes are saved and restored in bulk. When working in memory, nodes are saved and restored in named lists. Nodes are not actually saved until the end of the `tikzpicture` in which they are defined, meaning that if saving to memory then all the nodes in a `tikzpicture` will belong to the same list.

The keys for working with saving and restoring nodes are as follows.

- **save node**
  This is the key to put on a node that is to be saved.

- **set node group=<name>**
  Nodes are grouped together into a list that can be saved either to a file or for use later on in the document. This sets the name for the current group.

- **restore nodes from list=<name>**
  This restores the node information from the named list to the current `tikzpicture`. This is required both for when the node information comes from a file or from earlier in the same document.

- **save nodes to file**
  This is a `true/false` key which determines whether to save the node information to a file.

- **set saved nodes file name=<name>**
  This sets the file name for the saved nodes (the extension will be `.nodes`). The default is to use the current TeX filename. This is set globally, and once the file is opened then changing the name will have no effect. (The file is not opened until it is actually needed to avoid creating empty files unnecessarily.)

- **restore nodes from file=<name>**
  This loads the node information from the file into the current document. The `<name>` can have the syntax `[@options]{name}`, where `options` can be used to influence how the nodes are restored. The key `transform saved nodes` (see below) can be given here. Another useful key is the `name prefix` key which is applied to all restored nodes.

- **transform saved nodes**
A particular use-case for restoring saved nodes is to safely include one `tikzpicture` inside another by creating an image out of the inner picture and including it back in as a picture inside a node. In that situation, restoring the nodes from the inner picture can make it possible to refer to coordinates from the inner picture to the outer one. If there is a transformation in place on the containing node, this key applies that transformation to all the nodes in the inner picture.

5 Examples and Extras

The `\tikzmark` command has been used in numerous answers on [TeX-SX](https://tex.stackexchange.com). The plan is to gather some of these into extra libraries which can be loaded via `\usetikzmarklibrary`.

At present, this is the code listings library (which works with the `listings` package). One that is in development (as it has featured much on the TeX-SX website) is highlighting, however this is not so straightforward to implement so is still under development.

5.1 Basic Examples

A simple example of the `\tikzmark` macro is the following.

```latex
\[
\tikzmark{a} e^{i \pi/2} = i
\]
This\tikz[remember picture,overlay,baseline=0pt] \draw[->] (0,1em) to[bend left] ([shift={(-1ex,1ex)}]pic cs:a); is an important equation.

\begin{itemize}
\item A first item,\tikzmark{b}
\item A second item,\tikzmark{c}
\item A third item.\tikzmark{d}
\end{itemize}
\begin{tikzpicture}[remember picture,overlay]
\draw[decorate,decoration={brace}] ({pic cs:c} |- {pic cs:b}) +(0,1em) -- node[right,inner sep=1em] {some items} ({pic cs:c} |- {pic cs:d});
\end{tikzpicture}
```

```latex
\begin{itemize}
\item A first item,\tikzmark{b}
\item A second item,\tikzmark{c}
\item A third item.\tikzmark{d}
\end{itemize}
```
• A first item,
• A second item,
• A third item.

\begin{tikzpicture}[remember picture]
ode (a) at (0,0) {This has a \texttt{subnode}\{subnode\} in it};
\draw[->] (0,-1) to[bend right] (sub);
\end{tikzpicture}

This has a subnode in it

An example using \texttt{tikzmark} inside a \texttt{tikzpicture}

\begin{tikzpicture}
ode (a) at (0,0) {This has a \texttt{subnode}\{subnode\} in it};
\draw[->] (0,-1) to[bend right] (sub);
\end{tikzpicture}

By placing the \texttt{tikzmark} code before the marks, the arrow goes under the subsequent text and picture.

\begin{tikzpicture}illdraw[fill=gray] (0,0) circle[radius=1cm];
tikzmarknode{b}{(-1,-1)}
\end{tikzpicture}

By placing the code before the marks, the arrow goes under the subsequent text and picture.

The \texttt{tikzmarknode} puts a node around some text, which can be referred to later, and adds a \texttt{tikzmark} at its origin.
Putting a node around \tikzmarknode{txt}{some text} means we can connect text together, including in maths:
\[
\sum_{k=1}^n k^2 \tikzmarknode{b}{2}\]
\begin{tikzpicture}[remember picture,overlay]
\draw[->] (txt) -- (a);
\draw[->] (a.south) to[out=-90,in=-45] (b.south east);
\end{tikzpicture}

Putting a node around some text means we can connect text together, including in maths:
\[\sum_{k=1}^n k^2\]

The syntax for saving node data is illustrated by the following example.
File firstpicture.tex:
\begin{verbatim}
\documentclass[tikz,border=10pt]{standalone}
\usetikzlibrary{tikzmark,shapes.geometric}
\begin{document}
\begin{tikzpicture}[save nodes to file]
\node[draw,rotate=-30,save node](1) at (-2,0) {1};
\draw[->] (0,0) -- (1);
\node[draw,ellipse,save node] (c) at (current bounding box.center) {};
\end{tikzpicture}
\end{document}
\end{verbatim}

File secondpicture.tex:
\begin{verbatim}
\documentclass[tikz,border=10pt]{standalone}
\usetikzlibrary{tikzmark,shapes.geometric}
\begin{document}
\begin{tikzpicture}[save nodes to file]
\node[draw,rotate=-70,save node] (2) at (2,0) {2};
\draw[->] (0,0) -- (2);
\node[draw,ellipse,save node] (c) at (current bounding box.center) {};
\end{tikzpicture}
\end{document}
\end{verbatim}

Main file:
\documentclass[article]
\usepackage{tikz}
\usetikzlibrary{tikzmark}

\begin{document}
\begin{tikzpicture}
\node[draw, rotate=30, restore nodes from file={[transform saved nodes, name prefix=pic-1-]{firstpicture}}] (a-1) at (-2,-3) \includegraphics{firstpicture.pdf};
\node[draw, rotate=70, restore nodes from file={[transform saved nodes, name prefix=pic-2-]{secondpicture}}] (a-2) at (+2,+2) \includegraphics{secondpicture.pdf};
\draw[red] (pic-1-1.north west) -- (pic-1-1.north east) -- (pic-1-1.south east) -- (pic-1-1.south west) -- cycle;
\draw[red] (pic-2-2.north west) -- (pic-2-2.north east) -- (pic-2-2.south east) -- (pic-2-2.south west) -- cycle;
\node[red] at (pic-1-1) {1};
\node[red] at (pic-2-2) {2};
\draw (a-1) circle[radius=5pt];
\draw (a-2) circle[radius=5pt];
\draw (pic-1-1) -- (pic-2-2);
\end{tikzpicture}
\end{document}

This produces:
5.2 Code Listings

If the \texttt{listings} package has been loaded then issuing

\begin{verbatim}
\usetikzmarklibrary{listings}
\end{verbatim}

will load in some code to add marks to \texttt{lstlisting} environments. This code places a mark at three places on a line of code in a \texttt{listings} environment. The marks are placed at the start of the line, the first non-whitespace character, and the end of the line (if the line is blank the latter two are not placed). (This has not been extensively tested, it works by adding code to various “hooks” that are made available by the \texttt{listings} package; it is quite possible that the hooks chosen are both wrong and insufficient to cover all desired cases.)

These are inspired by questions such as \texttt{Marking lines in listings} and \texttt{Macros for code annotations}.

In more detail, the \texttt{listings} library places lots of marks around the code. The marks are:

- \texttt{line-<name>-<number>-start} at the start of each line.
- \texttt{line-<name>-<number>-end} at the end of each line.
- \texttt{line-<name>-<number>-first} at the first non-space character of the line (assuming it exists).

The line numbers \texttt{should} match up with the line numbers in the code in that any initial offset is also applied.
Not every mark is available on every line. If a line is blank, in particular, it will only have a start mark. The following example shows this, where the red dots are the start, the blue are end, and the green are first.

This example puts a fancy node behind certain lines of the code, computing the necessary extents.
# include <stdio.h>

int main(void)
{
    printf("hello, world\n");
    return 0;
}

\begin{lstlisting}[language=c, name=more code, numbers=left, firstnumber=3]

# include <stdio.h>

int main(void)
{
    printf("hello, world\n");
    return 0;
}
\end{lstlisting}

6 Acknowledgements

The \tikzmark macro has been used and abused by many users of \TeX-SX. Of particular note (but in no particular order) are Peter Grill, Gonzalo Medina, Claudio Fiandrino, percusse, and marmot. I would also like to mention David Carlisle whose knowledge of TikZ continues to astound us all.

7 Implementation

7.1 Main Code

The save nodes code uses \LaTeX3.

\begin{lstlisting}[language=TeX]
\RequirePackage{expl3, l3keys2e, xparse}
\tikzset{%
    remember picture with id/.style=,%
    remember picture,]
\end{lstlisting}

Not totally happy with using every picture here as it’s too easily overwritten by the user. Maybe it would be better to patch endtikzpicture directly.

\begin{lstlisting}[language=TeX]
every picture/.append style=,%
execute at end picture=%
\end{lstlisting}
The positions are already recorded in the aux file, all we really need to do is provide them with better names.

Provides a way to test if a picture has already been saved (in particular, can avoid errors on first runs)

Page handling

if tikzmark on current page/.code n args={3}{%
}
Prefix and suffix for tikzmark names, shamelessly borrowed from the main tikz code
\tikzmark prefix/.initial=,\%
\tikzmark suffix/.initial=,\%
%
\savepointas This is what gets written to the aux file.
\definesavereference#1#2#3#4{\}
\expandafter\edef\csname savepointas\endcsname#1{#2}
\expandafter\edef\csname savepointas\endcsname#1#3#4{#2}
\texttt{\textbackslash tikzmarkalias} Alias a tikzmark to another name (used in \texttt{tikzmarknode})
\begin{verbatim}
def\tikzmarkalias#1#2{%
  \pgf@node@gnamelet{save@pt@#1}{save@pt@#2}%
  \pgf@node@gnamelet{save@pt@#1@offset}{save@pt@#2@offset}%
}
\end{verbatim}

\texttt{\textbackslash tmk@labeldef} Auxiliary command for the coordinate system.
\begin{verbatim}
def\tmk@labeldef#1,#2\@nil{%
edf\tikzmark@pp@name{#1}%
def\tmk@def{#2}%
}
\end{verbatim}

\texttt{\texttt{pic}} This defines the new coordinate system.
\begin{verbatim}
tikzdeclarecoordinatesystem{pic}{%
  \pgfutil@in@,{#1}%
  \ifpgfutil@in@%
    \tmk@labeldef#1\@nil%
  \else
    \tmk@labeldef#1,(0pt,0pt)\@nil%
  \fi
  \@ifundefined{save@pt@\tmk@label}{%
    \pgfutil@firstofone\tmk@def%
  }{%}
  \pgf\sys\@getposition\{\cname save@pt@\tmk@label\@endcsname\}%
  \save@orig@pic%
  \pgf\sys\@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%
  \pgf@xa=\pgf@x%
  \pgf@ya=\pgf@y%
  \pgf\process%
  \pgf\process\{\pgfpointorigin\cname save@pt@\tmk@label@offset\@endcsname\%
  \advance\pgf@x by \pgf@x%
  \advance\pgf@y by \pgf@y%
  \@ifundefined{save@pg@\cname save@pt@\tmk@label\@endcsname}{%%
    \@ifundefined{save@pg@\pgfpictureid}{}{%
      \pgfkeysvalueof{/tikz/next page vector}%
      \edef\tmk@pg{\the\numexpr\csname save@pg@\tmk@label\@endcsname\endcsname - \csname save@pg@\pgfpictureid\endcsname\relax}%
      \ifnum \tmk@pg > 0 \relax%
        \advance \pgf@x by \pgf@x\relax%
        \advance \pgf@y by \pgf@y\relax%
      \fi%
      \ifnum \tmk@pg < 0 \relax%
        \advance \pgf@x by -\pgf@x\relax%
        \advance \pgf@y by -\pgf@y\relax%
      \fi%
      \advance \pgf@x by \pgf@x\relax%
      \advance \pgf@y by \pgf@y\relax%
    }
  }
\end{verbatim}
The active/non-active semi-colon is proving somewhat hazardous to \texttt{tikzmark} (see Tikzmark and french seem to conflict and Clash between tikzmark, babel package (french) and babel tikzlibrary) so \texttt{tikzmark} now uses the brace-delimited version of the \texttt{tikz} command.

This version is for when we’re outside a tikzpicture environment

\begin{verbatim}
\newcommand\tikzmark@outside[2][]{
    \tikz[remember picture with id=#2]{#1}
}
\end{verbatim}

This is for when we’re inside a tikzpicture environment

\begin{verbatim}
\def\tikzmark@inside#1#2{
    \tikzset{remember picture}\
    \tikz@scan@one@point\pgfutil@firstofone#2\relax\
    \protected@write\pgfutil@auxout{}{\string\savepointas{\tikzmark@pp@name{#1}}{\pgfpictureid}{\the\pgf@x}{\the\pgf@y}}\
}
\end{verbatim}

And finally, the ultimate invoker:

\begin{verbatim}
\def\tikzmark{
    \ifx\pgfpictureid\@undefined
        \let\tikzmark@next=\tikzmark@outside
    \else
        \relax
        \ifx\scope\tikz@origscope\relax
            \let\tikzmark@next=\tikzmark@outside
        \else
            \let\tikzmark@next=\tikzmark@inside
        \fi
    \fi
        \fi
    \fi
    \tikzmark@next
}
\end{verbatim}

\begin{verbatim}
\newcommand\pgfmark[1][]{
    \bgroup\
    \global\advance\pgf@picture@serial@count by1\relax\
    \edef\pgf@picture@serial@count@by1\relax\
    \edef\pgfpictureid{pgfid\the\pgf@picture@serial@count}\
    \pgfsys@markposition{\pgfpictureid}{\the\pgf@x}{\the\pgf@y}\relax\
    \protected@write\pgfutil@auxout{}{\string\savepointas{\tikzmark@pp@name{#1}}{\pgfpictureid}{\the\pgf@x}{\the\pgf@y}}\
}\end{verbatim}

19
If the beamer class is used, make the commands overlay aware.

\tikzmark<>\0ifclassloaded\{beamer\}{
\renewcommand<>\{\tikzmark@outside\}[2][]{%
\only#3\{\beameroriginal\{\tikzmark@outside\}{[#1]}{#2}\%
}\}
\renewcommand<>\{\tikzmark@inside\}[2]{%
\only#3\{\beameroriginal\{\tikzmark@inside\}{#1}{#2}\%
}\}
}

\pgfmark<>\0ifclassloaded\{beamer\}{
\renewcommand<>\{\pgfmark\}[1]{\only#2\{\beameroriginal\{\pgfmark\}{#1}\}}
}

If beamer is loaded, add a suffix based on the frame number

\0ifclassloaded\{beamer\}{
\tikzset{
\tikzmark suffix=-\the\beamer@slideinframe
}\}

\iftikzmark
\newif\iftikzmark@
\newcommand\iftikzmark[3][]{%
@ifundefined\{save@pt@\tikzmark@pp@name{#1}\}{%
 #3%
 #2%
}%
}%
A version suitable for \if ... \else ... \fi.

\newcommand\iftikzmarkexists[1][]{%
@ifundefined\{save@pt@\tikzmark@pp@name{#1}\}{%
 \tikzmark@false%
}{%
 \tikzmark@true%
}%
}\iftikzmark@
\iftikzmarkonpage
\newcommand\iftikzmarkonpage[2]{%  
@ifundefined{save@pt@\tikzmark@pp@name{#1}}{%  
  \tikzmark@false  
}{{%  
@ifundefined{save@pg@\csname save@pt@\tikzmark@pp@name{#1}\endcsname}{%  
  \tikzmark@false  
}{{%  
  \iftikzmark@  
}}}%
}
\iftikzmarkoncurrentpage
\newcommand\iftikzmarkoncurrentpage[1]{%  
@ifundefined{save@pt@\tikzmark@pp@name{#1}}{%  
  \tikzmark@false  
}{{%  
@ifundefined{save@pg@\csname save@pt@\tikzmark@pp@name{#1}\endcsname}{%  
  \tikzmark@false  
}{{%  
    \iftikzmark@  
  }}%}
}
\subnode{Note: much of this code was inevitably adapted from the node defining code in the TikZ/PGF sources.}
\def\subnode@#1#2#3{%  
  \begingroup  
  \pgfmark{#2}  
  \setbox\pgfnodeparttextbox\hbox\begingroup #3\egroup  
  \def\tikz@fig@name{#2}  
  \tikzset{every subnode/.try,#1}  
  \pgfpointorigin  
  \tikz@scan@one@point\pgfutil@firstofone(pic cs:#2)\relax  
  \advance\pgf@x by .5\wd\pgfnodeparttextbox  
  \advance\pgf@y by .5\ht\pgfnodeparttextbox  
  \advance\pgf@y by -.5\dp\pgfnodeparttextbox
}
\let\pgf@nodecallback\pgfutil@gobble
\def\tzmk@prfx{pgf@sys@pdf@mark@pos@pgfid}\
edef\tzmk@pic{\tzmk@prfx\the\pgf@picture@serial@count}\
\expandafter\ifx\csname\tzmk@pic\endcsname\relax\
\edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-1\relax}\
\expandafter\ifx\csname\tzmk@pic\endcsname\relax\
\edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-2\relax}\
\expandafter\ifx\csname\tzmk@pic\endcsname\relax\
\edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-3\relax}\
\expandafter\ifx\csname\tzmk@pic\endcsname\relax\
\pgfutil@ifundefined{pgf@sh@ns@#2}{\
\pgfnodealias{#2}{#2-t}\
\tikzmarkalias{#2}{#2-t}\
}\{}\
\else\
\pgfnodealias{#2}{#2-d}\
\tikzmarkalias{#2}{#2-d}\
\fi\
\else\
\pgfnodealias{#2}{#2-t}\
\tikzmarkalias{#2}{#2-t}\
\fi\
\else\
\pgfnodealias{#2}{#2-s}\
\tikzmarkalias{#2}{#2-s}\
\fi\
\else\
\pgfnodealias{#2}{#2-ss}\
\tikzmarkalias{#2}{#2-ss}\
\fi\
\subnode@{#1}{#2}{#3}\
}

\tikzmarknode The \texttt{tikzmark} macro has changed considerably since its first inception, but there does still seem to be a use for the original version which put stuff inside a node. This command reintroduces that command.

It does its best to work inside a math environment by a sneaky trick involving \texttt{\mathchoice}: the \texttt{remember picture} key means that only the picture id of the typeset box is saved to the aux file. So comparing the possible picture ids of the four options with the one read from the aux file, we can figure out which box was actually used.

\def\tikzmarknode@#1#2#3{\
\tikz[\
\remember picture,\
\baseline=(#2.base),\
\save picture id={#2},\
\]
every tikzmarknode picture/.try
\node[anchor=base, inner sep=0pt, minimum width=0pt, name={#2}, node contents={#3}, every tikzmarknode/.try, #1]
}\)
}

\newcommand\tikzmarknode[3]{%
\ifmmode
\mathchoice{%\tikzmarknode@{#1}{#2-d}{\(\displaystyle #3\)}%}{%\tikzmarknode@{#1}{#2-t}{\(\textstyle #3\)}%}{%\tikzmarknode@{#1}{#2-s}{\(\scriptstyle #3\)}%}{%\tikzmarknode@{#1}{#2-ss}{\(\scriptscriptstyle #3\)}%}%
\let\pgf@nodecallback\pgfutil@gobble
\def\tzmk@prfx{pgf@sys@pdf@mark@pos@pgfid}%
\edef\tzmk@pic{\tzmk@prfx\the\pgf@picture@serial@count}%
\expandafter\ifx\csname\tzmk@pic\endcsname\relax
\edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-1\relax}%
\expandafter\ifx\csname\tzmk@pic\endcsname\relax
\edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-2\relax}%
\expandafter\ifx\csname\tzmk@pic\endcsname\relax
\edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-3\relax}%
\expandafter\ifx\csname\tzmk@pic\endcsname\relax
\pgfutil@ifundefined{pgf@sh@ns@#2}{%\pgfnodealias{#2}{#2-t} \tikzmarkalias{#2}{#2-t} \relax}{%\pgfnodealias{#2}{#2-d} \tikzmarkalias{#2}{#2-d} \relax}{%\pgfnodealias{#2}{#2-s} \tikzmarkalias{#2}{#2-s} \relax}{%\relax}
\else
\pgfnodealias{#2}{#2-t} \tikzmarkalias{#2}{#2-t} \relax
\else
\pgfnodealias{#2}{#2-s} \tikzmarkalias{#2}{#2-s} \relax
\fi
\else
\pgfnodealias{#2}{#2-t} \tikzmarkalias{#2}{#2-t} \relax
\else
\pgfnodealias{#2}{#2-s} \tikzmarkalias{#2}{#2-s} \relax
\fi
\relax}
The save node code is written in \LaTeX3.

\ExplSyntaxOn

We save our information in a “property list”, which is L3’s version of an associative array or dictionary. They keys will give the ability to store several groups of nodes and restore them at will.

\prop_new:N \g__sn_prop

We’ll need a couple of spare token lists

\tl_new:N \l__sn_tmpa_tl
\tl_new:N \l__sn_tmpb_tl
Another useful token list

438 \tl_new:N \l__open_bracket_tl
439 \tl_set:Nn \l__open_bracket_tl {\[}\]

This token list is used for our current node group name

440 \tl_new:N \l__sn_group_tl

We store up the nodes in a list and save them at the end of a given tikzpicture.

441 \clist_new:N \l__sn_nodes_clist

This boolean is for whether we save to a file or not.

442 \bool_new:N \l__sn_file_bool

This boolean is for whether we are in the preamble or not.

443 \bool_new:N \g__sn_preamble_bool
444 \bool_set_true:N \g__sn_preamble_bool

445 \msg_new:nnn {tikzmark} {no file} {File~ "#1"~ doesn't~ exist.}
446 \msg_new:nnn {tikzmark} {loading nodes} {Loading~ nodes~ from~ "#1".}

Dimensions and token lists for shifting

447 \dim_new:N \l__sn_x_dim
448 \dim_new:N \l__sn_y_dim
449 \dim_new:N \l__sn_xa_dim
450 \dim_new:N \l__sn_ya_dim
451 \tl_new:N \l__sn_centre_tl
452
453 \tl_new:N \l__sn_transformation_tl
454 \tl_set:Nn \l__sn_transformation_tl {{1}{0}{0}{1}{0pt}{0pt}}

Set up a stream for saving the nodes data to a file

455 \iow_new:N \g__sn_stream
456 \bool_new:N \g__sn_stream_bool
457 \tl_new:N \g__sn_filename_tl
458 \tl_gset:Nx \g__sn_filename_tl {\c_sys_jobname_str}
459
460 \cs_new_nopar:Npn \sn_open_stream: {
461 \bool_if:NT \g__sn_stream_bool
462 { \iow_open:Nn \g__sn_stream {\tl_use:N \g__sn_filename_tl .nodes}
463 \bool_gset_true:N \g__sn_stream_bool
464 }
465 }
466
467 \AtEndDocument
468 {
469 \ExplSyntaxOn
470 \bool_if:NT \g__sn_stream_bool
471 { \iow_close:N \g__sn_stream
472 }
473 }
474
475
26
LaTeX3 wrappers around some PGF functions (to avoid \@-catcode issues)

\makeatletter
\cs_set_eq:NN \tikz_set_node_name:n \tikz@pp@name
\cs_set_eq:NN \tikz_fig_must_be_named: \tikz@fig@mustbenamed
\makeatother

This is the command that actually does the work. It constructs a token list which contains the code that will restore the node data when invoked. The two arguments are the token list to store this in and a comma separated list of the node names to be saved.

\save_nodes:Nn
\save_nodes:Nn #1 #2
\makeatletter
\cs_generate_variant:Nn \tikz_scan_point:n {V}
\cs_generate_variant:Nn \tikz_scan_point:NNn {NNV}
\save_nodes:Nn
\makeatother
\cs_generate_variant:Nn \tikz_scan_point:n {V}
\cs_generate_variant:Nn \tikz_scan_point:NNn {NNV}

Clear our token lists
\tl_clear:N \l__sn_tmpa_tl

Set the centre of the picture
\tl_clear:N \l__sn_tmpa_tl
\tikz_set:NNn \l__sn_x_dim \l__sn_y_dim {{current- bounding- box.center}}
\dim_set:Nn \l__sn_x_dim {\l__sn_x_dim}
\dim_set:Nn \l__sn_y_dim {\l__sn_y_dim}
\tl_set:Nx \l__sn_centre_tl {1}{0}{0}{1}{{dim_use:N \l__sn_x_dim}{dim_use:N \l__sn_y_dim}}
\clist_map_inline:nn {#2}
\tl_if_exist:cT {pgf@sh@ns@##1}
\pgf@sh@XX@nodename

The node information is stored in a series of macros of the form \pgf@sh@XX@nodename where XX is one of the following.

27
Our token list will look like:
\begin{verbatim}
\tl_set:cn {pgf@sh@XX@nodename} <current contents of that macro>
\end{verbatim}
This will restore \texttt{pgf@sh@XX@nodename} to its current value when this list is
invoked.

This part puts the \texttt{\tl_set:cn \{pgf@sh@XX@nodename\}} in place
\begin{verbatim}
\tl_put_right:Nn \l__sn_tmpa_tl \tl_gset:cn {pgf@sh@####1@ \tikz_set_node_name:n{##1} } \}
\end{verbatim}
Now we put the current contents in place. We’re doing this in an expansive
context to get at the contents. The \texttt{\exp_not:v} part takes the current value of
\texttt{pgf@sh@XX@nodename} and puts it in place, preventing further expansion.
\begin{verbatim}
\tl_if_exist:cTF {pgf@sh@####1@##1} {
\tl_put_right:Nx \l__sn_tmpa_tl {{\exp_not:v \{pgf@sh@####1@ \tikz_set_node_name:n{##1}\}}} 
\}
\}
\tl_put_right:Nx \l__sn_tmpa_tl {{}}
\}
\tl_put_right:Nn \l__sn_tmpa_tl \tl_gset:cn {pgf@sh@nt@ \tikz_set_node_name:n{##1} }
\compose_transformations:NVv \l__sn_transformation_tl \l__sn_centre_tl {pgf@sh@nt@##1}
\tl_put_right:Nx \l__sn_tmpa_tl {{\exp_not:v \l__sn_tmpb_tl}}
\tl_put_right:Nn \l__sn_tmpa_tl { \}
\transform_node:Nn \l__sn_transformation_tl { \}
\tikz_set_node_name:n{##1}
\)
\}
\}
Once we’ve assembled our token list, we store it in the given token list
\begin{verbatim}
\tl_set_eq:NN #1 \l__sn_tmpa_tl
\end{verbatim}
\save_nodes_to_list:nn Save the nodes to a list, given a key
\begin{verbatim}
\cs_new_nopar:Npn \save_nodes_to_list:nn #1#2 \{
\save_nodes:Nn \l__sn_tmpa_tl \g__sn_prop {#1} \l__sn_tmpa_tl
\}
\end{verbatim}
Save the nodes to a file
\save_nodes_to_file:n \cs_new_nopar:Npn \save_nodes_to_file:n #1
\save_nodes:Nn \l__sn_tmpa_tl {#1}
Save the token list to the nodes file so that on reading it back in, we restore the node definitions
\sn_open_stream:
\iow_now:Nx \g__sn_stream
{
\iow_newline:
\exp_not:V \l__sn_tmpa_tl
\iow_newline:
}
\cs_generate_variant:Nn \save_nodes_to_list:nn {VV}
\cs_generate_variant:Nn \save_nodes_to_file:n {V}

Restoring nodes is simple: look in the property list for the key and if it exists, invoke the macro stored there.
\prop_get:NnNT \g__sn_prop {#1} \l__sn_tmpa_tl
{
\tl_use:N \l__sn_tmpa_tl
}

Restoring nodes from file:
\restore_nodes_from_file:n \cs_new_nopar:Npn \restore_nodes_from_file:n #1
\file_if_exist:nTF {#1.nodes}
{
\msg_log:nnn {tikzmark} {loading nodes} {#1}
\ExplSyntaxOn
\file_input:n {#1.nodes}
\ExplSyntaxOff
}
{
\msg_warning:nnn {tikzmark} {no file} {#1}
}
\AtBeginDocument{\bool_gset_false:N \g__sn_preamble_bool}

Compose PGF transformations \#2 \times \#3, storing the result in \#1
\compose_transformations:Nnn
I think the PGF Manual might be incorrect. It implies that the matrix is stored row-major, but experimentation implies column-major.
That is, \{a\} \{b\} \{c\} \{d\} \{s\} \{t\} is:

\[
\begin{bmatrix}
a & c \\
b & d
\end{bmatrix}
\]

\cs_new_nopar:Npn \compose_transformations:Nnn #1#2#3
\tl_gset:Nx #1
\{\fp_eval:n {
\tl_item:nn {#2} {1} \\
* \tl_item:nn {#3} {1} \\
+ \\
\tl_item:nn {#2} {3} \\
* \tl_item:nn {#3} {2}
}
\}
\{\fp_eval:n {
\tl_item:nn {#2} {2} \\
* \tl_item:nn {#3} {1} \\
+ \\
\tl_item:nn {#2} {4} \\
* \tl_item:nn {#3} {2}
}
\}
\{\fp_eval:n {
\tl_item:nn {#2} {1} \\
* \tl_item:nn {#3} {3} \\
+ \\
\tl_item:nn {#2} {3} \\
* \tl_item:nn {#3} {4}
}
\}
\{\fp_eval:n {
\tl_item:nn {#2} {2} \\
* \tl_item:nn {#3} {3} \\
+ \\
\tl_item:nn {#2} {4} \\
* \tl_item:nn {#3} {4}
}
\}
\{\fp_to_dim:n {
\tl_item:nn {#2} {1} \\
* \tl_item:nn {#3} {5} \\
+ \\
\tl_item:nn {#2} {3} \\
* \tl_item:nn {#3} {6} \\
+ \\
\tl_item:nn {#2} {5}
}
\}
\maybe_save_nodes:
}

Are we saving to a file?
\save_nodes_to_file/.code={
\tl_if_eq:nnTF {#1}{false}{
\bool_set_false:N \l__sn_file_bool
}
{
\bool_set_true:N \l__sn_file_bool
}

\pgfkeysalso{
execute_at_end_scope={
\maybe_save_nodes:
}}

Append current node to the list of nodes to be saved
\save_node/.code=
\tikz_fig_must_be_named:
\pgfkeysalso{append_after_command={
\pgfextra{
\clist_gput_right:Nv \l__sn_nodes_clist {tikz@last@fig@name}
}\pgfextra{
\scope
\split_argument:NNn \tikzset \restore_nodes_from_file:n {#1}
\endscope
}}}\

Restore nodes from file
\restore_nodes_from_file/.code={
\bool_if:NTF \g__sn_preamble_bool {
\tl_if_equal:TTF \tikzset \restore_nodes_from_file:n {\g__sn_filename_tl}
}
{
\tikz_fig_must_be_named:
\pgfkeysalso{append_after_command={
\pgfextra{
\scope
\split_argument:NNn \tikzset \restore_nodes_from_file:n {#1}
\endscope
}}}\

\restore_nodes_from_file/.default = \g__sn_filename_tl,
Restore nodes from list

```
\restore_nodes_from_list/.code={
\tikz_fig_must_be_named:
\pgfkeysalso{append_after_command={
\pgfextra{
\scope
\split_argument:NNn \tikzset \restore_nodes_from_list:n {#1}
\endscope}
}
\maybe_save_nodes:
```

\split_argument:Nnn

```
\cs_new_nopar:Nnn \split_argument:Nnn #1#2#3
{
\tl_set:Nx \l__sn_tmpa_tl {\tl_head:n {#3}}
\tl_if_eq:NNTF \l__sn_tmpa_tl \l__open_bracket_tl
{\split_argument_aux:Nnp \l__sn_group_tl #1#2#3
}{#2 {#3}}
}
```

\split_argument_aux:Nnp

```
\cs_new_nopar:Nnn \split_argument_aux:Nnp #1#2[#3]#4
{
\clist_if_empty:NFN \l__sn_nodes_clist
{\bool_if:NTF \l__sn_file_bool
\{\save_nodes_to_file:V \l__sn_nodes_clist
\}
\{\tl_if_empty:NFN \l__sn_group_tl
\{\save_nodes_to_list:VV \l__sn_group_tl \l__sn_nodes_clist
\}
```

\maybe_save_nodes:
7.2 Listings

From [http://tex.stackexchange.com/q/79762/86](http://tex.stackexchange.com/q/79762/86)

\iflst@linemark
\newif\iflst@linemark
\ExplSyntaxOff

\ifpackageloaded{listings}{%

A conditional to help with placing the mark at the first non-whitespace character.
\everyline
This hook places the mark at the start of the line.
\begingroup
\advance\c@lstnumber by 1\relax
\pgfmark{line-\lst@name-\the\c@lstnumber-start}\
\endgroup

EOL
This hook places the mark at the end of the line and resets the conditional for
placing the first mark.
\lst@AddToHook{EOL}{\pgfmark{line-\lst@name-\the\c@lstnumber-end}\
\global\lst@linemarktrue
\}

OutputBox
Experimenting shows that this is the right place to set the mark at the first non-
whitespace character. But we only want to do this once per line.
\lst@AddToHook{OutputBox}{%\iflst@linemark
\pgfmark{line-\lst@name-\the\c@lstnumber-first}\
\global\lst@linemarkfalse
\fi
\}

\tikzmk@lst@fnum
An auxiliary macro to figure out if the \texttt{firstnumber} key was set. If so, it has the
form \texttt{<number>}/\relax. If not, it expands to a single token.
\def\tikzmk@lst@fnum#1\relax#2\@STOP{%\def\@test{#2}\
\ifx\@test\@empty
\def\tkzmk@lst@start{0}\
\else
\@tempcnta=#1\relax
\advance\@tempcnta by -1\relax
\def\tkzmk@lst@start{\the\@tempcnta}\
\fi
\}
\ExplSyntaxOn
\endinput
\lst@AddToHook{Init}{%
\expandafter\tkzmk@lst@fnum\lst@firstnumber\relax\@STOP
\pgfmark{line-\lst@name-\tkzmk@lst@start-start}%
}%
\PackageError{tikzmark listings}{The listings package has not been loaded.}{}