The tikzmark package

Andrew Stacey
loopspace@mathforge.org

v1.3 from 2017/06/01

1 Introduction

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

```
\usepackage{tikz}
\usetikzlibrary{tikzmark}
```

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

```
\tikz[remember picture] \draw[overlay] (0,0) -- (pic
   cs:<name>);
```

There are two important points to note:

1. The enveloping `\tikz` or `tikzpicture` must have the key `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 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:

```
\newcommand{\tikzmark}[1]{\tikz[overlay,remember picture]
    \node (#1) {};}
```

Its first use was in the (inelegant) code:

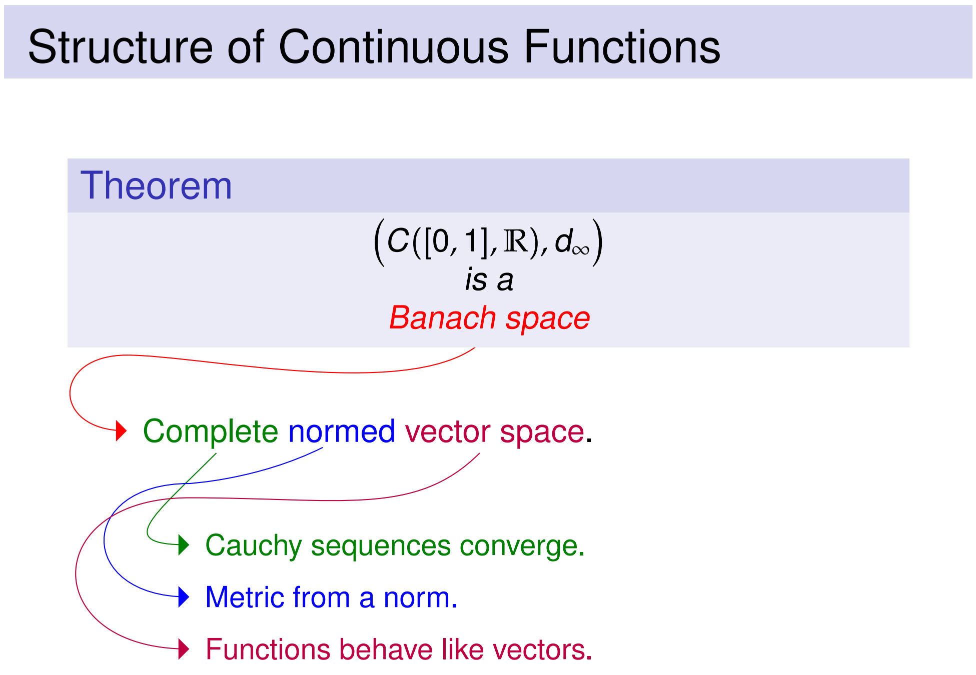
```
\begin{frame}
\frametitle{Structure of Continuous Functions}

\begin{tikzpicture}[overlay, remember picture]
\useasboundingbox (0,0);
\draw<2-|trans: 0|handout: 0>[red,->] (bsp) .. controls
   +(-1,-1) and ($(cnvs.north)+(1,1)$) ..
    ($(cnvs.north)+(0,1)$) .. controls
    ($(cnvs.north)+(-1,1)$) and +(-1,0) .. (cnvs.north);
\draw<3-|trans: 0|handout: 0>[green!50!black,->] (cplt) ..
    controls +(-1,-1) and +(-1,0) .. (mcplt.north);
\draw<4-|trans: 0|handout: 0>[blue,->] (norm) .. controls
   +(-1,-.5) and ($(nvs.north)+(0,1.5)$) ..
    ($(nvs.north)+(0,1.5)$) .. controls
    ($(nvs.north)+(-1.5,1.5)$) and +(-1.5,0) ..
    (nvs.north);
\draw<5-|trans: 0|handout: 0>[purple,->] (vector) ..
    controls +(-1,-1) and ($(vsp.north)+(2,2)$) ..
    ($(vsp.north)+(0,2)$) .. controls
    ($(vsp.north)+(-2,2)$) and +(-2,0) .. (vsp.north);
\end{tikzpicture}

\begin{theorem}
\centering
\(\big(C([0,1],\R),d_\infty\big)\) \\
is a \\
\alert{Banach\tikzmark{bsp} space}
\end{theorem}

\pause
\bigskip

\begin{itemize}
\item[\tikzmark{cnvs}]
    {\color<.(2)->{green!50!black}Comp\tikzmark{cplt}lete}
    {\color<.(3)->{blue}nor\tikzmark{norm}med}
    {\color<.(4)->{purple}vector\tikzmark{vector} space}.

\bigskip
\bigskip
\pause

\begin{itemize}[<+->]
\item[\tikzmark{mcplt}] {\color{green!50!black}Cauchy
    sequences converge.}
\medskip
\item[\tikzmark{nvs}] {\color{blue}Metric from a norm.}
\medskip
\item[\tikzmark{vsp}] {\color{purple}Functions behave like
    vectors.}
\end{itemize}
\end{itemize}

\end{frame}
```

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



Structure of Continuous Functions

Theorem

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

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

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.

4 Usage

This package defines the following commands and usable stuff.

1. `\tikzmark[`⟨*drawing command*⟩`]{`⟨*name*⟩`}`

   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 ... ;` 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`.

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

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

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

5. `(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).

6. `/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.

7. `/tikz/if picture id=#1#2#3`

   This is a key equivalent of the `\iftikzmark` command.

8. `/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.)

9. `/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 at 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 `\tikzmark`s on them, and the positions of the `\tikzmark`s 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.

10. `\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.

5 Examples and Extras

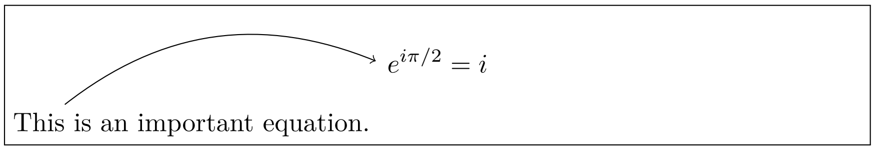
The `\tikzmark` command has been used in numerous answers on TeX-SX. The plan is to gather some of these into extra libraries which can be loaded via `\usetikzmarklibrary`.

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.

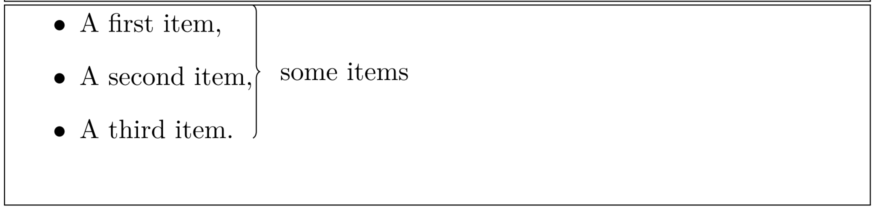
5.1 Basic Examples

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

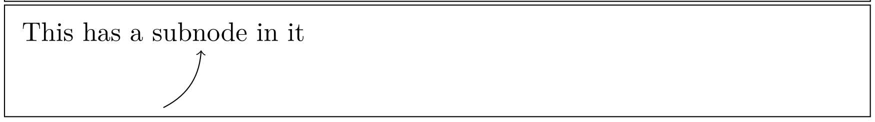
```
\[
 \tikzmark{a} e^{i \pi/2} = i
\]

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}
```



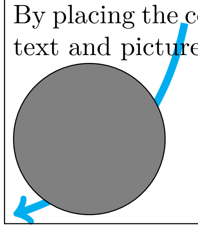
```
\begin{tikzpicture}[remember picture]
\node (a) at (0,0) {This has a \subnode{sub}{subnode} in
    it};
\draw[->] (0,-1) to[bend right] (sub);
\end{tikzpicture}
```



An example using `\tikzmark` inside a `tikzpicture`

```
\tikzset{tikzmark prefix=ex3-}
\begin{tikzpicture}[remember picture,overlay]
\draw[->,line width=1mm,cyan] (pic cs:a) to[bend left]
    (pic cs:b);
\end{tikzpicture}

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

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

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



5.2 Code Listings

If the `listings` package has been loaded then issuing `\usetikzmarklibrary{listings}` will load in some code to add marks to `lstlisting` environments. This code places a mark at three places on a line of code in a `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 `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 Marking lines in listings and Macros for code annotations.

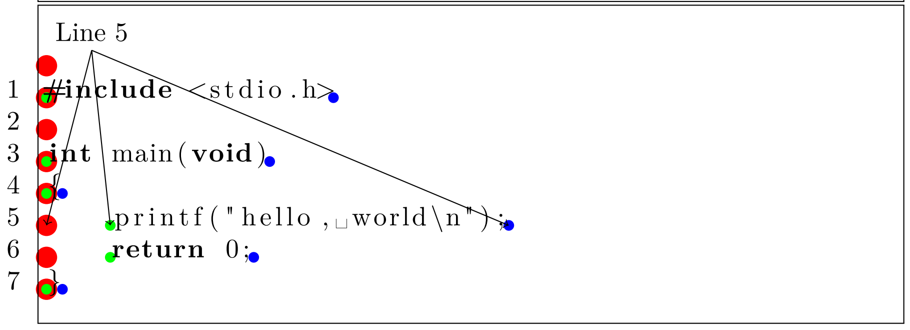
In more detail, the `listings` library places lots of marks around the code. The marks are:

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

The line numbers *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`.

```
\begin{tikzpicture}[remember picture]
\foreach \k in {0,...,7} {
\iftikzmark{line-code-\k-start}{\fill[red,overlay] (pic
    cs:line-code-\k-start)
    circle[radius=4pt];}{\message{No start for \k}}
\iftikzmark{line-code-\k-end}{\fill[blue,overlay] (pic
    cs:line-code-\k-end)
    circle[radius=2pt];}{\message{No end for \k}}
\iftikzmark{line-code-\k-first}{\fill[green,overlay]
    (pic cs:line-code-\k-first)
    circle[radius=2pt];}{\message{No first for \k}}
}
\draw[->,overlay] (0,0) -- (pic cs:line-code-5-first);
\draw[->,overlay] (0,0) -- (pic cs:line-code-5-start);
\draw[->,overlay] (0,0) -- (pic cs:line-code-5-end);
\node[above] at (0,0) {Line 5};
\end{tikzpicture}

\begin{lstlisting}[language=c,name=code,numbers=left]
  #include <stdio.h>

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



```
#include <stdio.h>

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

This example puts a fancy node behind certain lines of the code, computing the necessary extents.

```
\balloon{comment}{more code}{3}{3}
\balloon{comment}{more code}{7}{8}
\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}
```

```
3 #include <stdio.h>
4
5 int main(void)
6 {
7     printf("hello,␣world\n");
8     return 0;
9 }
```

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, and percusse. I would also like to mention David Carlisle whose knowledge of TikZ continues to astound us all.

7 Implementation

7.1 Main Code

```
\tikzset{%
  remember picture with id/.style={%
    remember picture,
    overlay,
    save picture id=#1,
  },
```

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.

```
  every picture/.append style={%
    execute at end picture={%
      \ifpgfrememberpicturepositiononpage%
      \edef\pgf@temp{%
```

```
        \noexpand\write\noexpand\pgfutil@auxout{%
        \string\savepicturepage{\pgfpictureid}{\noexpand\arabic{page}}}}%
      \pgf@temp
      \fi%
    },
  },
```

The positions are already recorded in the aux file, all we really need to do is provide them with better names.

```
  save picture id/.code={%
      \protected@write\pgfutil@auxout{}{%
        \string\savepointas{\tikzmark@pp@name{#1}}{\pgfpictureid}{0pt}{0pt}}%
  },
```

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

```
  if picture id/.code args={#1#2#3}{%
    \@ifundefined{save@pt@\tikzmark@pp@name{#1}}{%
      \pgfkeysalso{#3}%
    }{
      \pgfkeysalso{#2}%
    }
  },
```

Page handling

```
  next page/.is choice,
  next page vector/.initial={\pgfqpoint{0pt}{0pt}},
  next page/below/.style={%
    next page vector={\pgfqpoint{0pt}{-\the\paperheight}}%
  },
  next page/above/.style={%
    next page vector={\pgfqpoint{0pt}{\the\paperheight}}%
  },
  next page/left/.style={%
    next page vector={\pgfqpoint{-\the\paperwidth}{0pt}}%
  },
  next page/right/.style={%
    next page vector={\pgfqpoint{\the\paperwidth}{0pt}}%
  },
  next page/ignore/.style={%
    next page vector={\pgfqpoint{0pt}{0pt}}%
  },
```

Prefix and suffix for tikzmark names, shamelessly borrowed from the main tikz code

```
  tikzmark prefix/.initial=,%
  tikzmark suffix/.initial=,%
}
```

\tikzmark@pp@name

```
\def\tikzmark@pp@name#1{\csname pgfk@/tikz/tikzmark prefix\endcsname#1\csname pgfk@/tikz/tikzma
```

\savepointas This is what gets written to the aux file.

```
\def\savepointas#1#2#3#4{%
  \expandafter\gdef\csname save@pt@#1\endcsname{#2}%
  \expandafter\gdef\csname save@pt@#1@offset\endcsname{\pgfqpoint{#3}{#4}}%
}
\def\savepicturepage#1#2{%
  \expandafter\gdef\csname save@pg@#1\endcsname{#2}%
}
```

`\tmk@labeldef` Auxiliary command for the coordinate system.

```
\def\tmk@labeldef#1,#2\@nil{%
  \edef\tmk@label{\tikzmark@pp@name{#1}}%
  \def\tmk@def{#2}%
}
```

`pic` This defines the new coordinate system.

```
\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
      \pgf@xa=\pgf@x
      \pgf@ya=\pgf@y
    \pgf@process{\pgfpointorigin\csname save@pt@\tmk@label @offset\endcsname}%
    \advance\pgf@xa by \pgf@x
    \advance\pgf@ya by \pgf@y
      \@ifundefined{save@pg@\csname save@pt@\tmk@label\endcsname}{}{%
        \@ifundefined{save@pg@\pgfpictureid}{}{%
          \pgfkeysvalueof{/tikz/next page vector}%
          \advance \pgf@xa by \csname save@pg@\csname save@pt@\tmk@label\endcsname\endcsname\pg
\advance \pgf@ya by \csname save@pg@\csname save@pt@\tmk@label\endcsname\endcsname\pgf@y\relax
          \advance \pgf@xa by -\csname save@pg@\pgfpictureid\endcsname\pgf@x\relax
\advance \pgf@ya by -\csname save@pg@\pgfpictureid\endcsname\pgf@y\relax
        }%
      }%
      \pgf@x=\pgf@xa
      \pgf@y=\pgf@ya
    \pgftransforminvert
    \pgf@pos@transform{\pgf@x}{\pgf@y}%
    }%
  }
```

`\tikzmark` The active/non-active semi-colon is proving somewhat hazardous to `\tikzmark` (see Tikzmark and french seem to conflict and Clash between tikzmark, babel package (french) and babel tikzlibrary) so `\tikzmark` now uses the brace-delimited version of the `\tikz` command.

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

```
\newcommand\tikzmark@outside[2][]{%
\tikz[remember picture with id=#2]{#1}%
}
```

This is for when we're inside a tikzpicture environment

```
\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 pt}{\the\pgf@y pt}}%
}
```

And finally, the ultimate invoker:

```
\def\tikzmark{%
  \ifx\pgfpictureid\@undefined
  \let\tikzmark@next=\tikzmark@outside
  \else
  \let\tikzmark@next=\tikzmark@inside
  \fi
  \tikzmark@next%
}
```

`\pgfmark`

```
\newcommand\pgfmark[1]{%
    \bgroup
    \global\advance\pgf@picture@serial@count by1\relax%
    \edef\pgfpictureid{pgfid\the\pgf@picture@serial@count}%
      \pgfsys@markposition{\pgfpictureid}%
    \edef\pgf@temp{%
      \noexpand\write\noexpand\pgfutil@auxout{%
        \string\savepicturepage{\pgfpictureid}{\noexpand\arabic{page}}}}%
    \pgf@temp
    \protected@write\pgfutil@auxout{}{%
      \string\savepointas{\tikzmark@pp@name{#1}}{\pgfpictureid}{0pt}{0pt}}%
    \egroup
  }
```

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

`\tikzmark<>`

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

`\pgfmark<>`

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

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

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

`\iftikzmark`

```
\newcommand\iftikzmark[3]{%
    \@ifundefined{save@pt@#1}{%
      #3%
    }{%
      #2%
    }%
  }%
```

`\subnode`

```
\newcommand\subnode[3][]{%
  \begingroup
  \pgfmark{#2}%
  \setbox\pgfnodeparttextbox=\hbox\bgroup #3\egroup
  \def\tikz@shape{rectangle}%
  \def\tikz@anchor{center}%
  \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
  \pgftransformshift{}%
  \setbox\@tempboxa=\hbox\bgroup
  \pgfutil@ifundefined{pgf@sh@s@\tikz@shape}%
  {\PackageError{pgf}{Unknown shape ‘‘\tikz@shape’’}{}}%
  {%
    {%
      \let\pgf@sh@savedmacros=\pgfutil@empty% MW
      \let\pgf@sh@savedpoints=\pgfutil@empty%
      \def\pgf@sm@shape@name{\tikz@shape}% CJ % TT added prefix!
      \csname pgf@sh@s@\tikz@shape\endcsname%
      \pgf@sh@savedpoints%
      \pgf@sh@savedmacros% MW
      \pgftransformshift{%
        \pgf@sh@reanchor{\tikz@shape}{\tikz@anchor}%
        \pgf@x=-\pgf@x%
        \pgf@y=-\pgf@y%
      }%
      \expandafter\pgfsavepgf@process\csname pgf@sh@sa@\tikz@fig@name\endcsname{%
       \pgf@sh@reanchor{\tikz@shape}{\tikz@anchor}% FIXME : this is double work!
      }%
      % Save the saved points and the transformation matrix
      \edef\pgf@node@name{\tikz@fig@name}%
      \ifx\pgf@node@name\pgfutil@empty%
      \else%
        \expandafter\xdef\csname pgf@sh@ns@\pgf@node@name\endcsname{\tikz@shape}%
        \edef\pgf@sh@@temp{\noexpand\gdef\expandafter\noexpand\csname pgf@sh@np@\pgf@node@name\
        \expandafter\pgf@sh@@temp\expandafter{\pgf@sh@savedpoints}%
        \edef\pgf@sh@@temp{\noexpand\gdef\expandafter\noexpand\csname pgf@sh@ma@\pgf@node@name\
        \expandafter\pgf@sh@@temp\expandafter{\pgf@sh@savedmacros}% MW
        \pgfgettransform\pgf@temp
        \expandafter\xdef\csname pgf@sh@nt@\pgf@node@name\endcsname{\pgf@temp}%
        \expandafter\xdef\csname pgf@sh@pi@\pgf@node@name\endcsname{\pgfpictureid}%
      \fi%
    }%
  }%
  \egroup
  \box\pgfnodeparttextbox
  \endgroup
}
```

\usetikzmarklibrary

```
\def\usetikzmarklibrary{\pgfutil@ifnextchar[{\use@tikzmarklibrary}{\use@@tikzmarklibrary}}%}
\def\use@tikzmarklibrary[#1]{\use@@tikzmarklibrary{#1}}
\def\use@@tikzmarklibrary#1{%
  \edef\pgf@list{#1}%
  \pgfutil@for\pgf@temp:=\pgf@list\do{%
    \expandafter\pgfkeys@spdef\expandafter\pgf@temp\expandafter{\pgf@temp}%
    \ifx\pgf@temp\pgfutil@empty
    \else
      \expandafter\ifx\csname tikzmark@library@\pgf@temp @loaded\endcsname\relax%
      \expandafter\global\expandafter\let\csname tikzmark@library@\pgf@temp @loaded\endcsname=\
      \expandafter\edef\csname tikzmark@library@#1@atcode\endcsname{\the\catcode`\@}
      \expandafter\edef\csname tikzmark@library@#1@barcode\endcsname{\the\catcode`\|}
      \catcode`\@=11
      \catcode`\|=12
      \pgfutil@InputIfFileExists{tikzmarklibrary\pgf@temp.code.tex}{}{
          \PackageError{tikzmark}{I did not find the tikzmark extras library '\pgf@temp'.}{}
        }%
      \catcode`\@=\csname tikzmark@library@#1@atcode\endcsname
      \catcode`\|=\csname tikzmark@library@#1@barcode\endcsname
      \fi%
    \fi
  }%
}

```

7.2 Listings

From http://tex.stackexchange.com/q/79762/86

```
\@ifpackageloaded{listings}{%
```

`\iflst@linemark` A conditional to help with placing the mark at the first non-whitespace character.

```
    \newif\iflst@linemark
```

`EveryLine` This hook places the mark at the start of the line.

```
\lst@AddToHook{EveryLine}{%
  \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 `firstnumber` key was set. If so, it has the form `<number>\relax`. If not, it expands to a single token.

```
\def\tkzmk@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
}
```

`Init` Adds a mark at the start of the listings environment.

```
\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.}{}
}
```