

The `tikzmark` package

Andrew Stacey
`loopspace@mathforge.org`

v1.7 from 2019/05/07

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],\mathbb{R}),d_\infty\big)\bigg) \bigg) \bigg)
```

```

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

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

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[<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`. 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.)
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 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 `\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. The plan is to gather some of these into extra libraries which can be loaded via `\usetikzmarklibrary`.

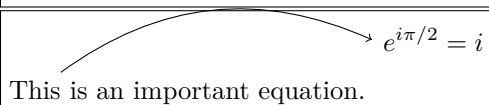
At present, this is the code listings library (which works with the `listings` package). One that is in development (as it has featured much on the TeX-SX website) is highlighting, however this is not so straightforward to implement so is still under development.

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.
```



This 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}
```

- A first item,
 - A second item,
 - A third item.
- } some items

```
\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}
```

This has a subnode in it

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}

```

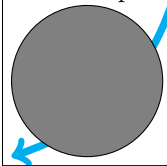
By placing the `\tikzmark{a}` code before the marks, the arrow goes under the subsequent text and picture.

```

\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.



The `\tikzmarknode` puts a node around some text, which can be referred to later, and adds a `\tikzmark` at its origin.

```

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

\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`:

```

\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}

```

File `secondpicture.tex`:

```

\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}

```

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;

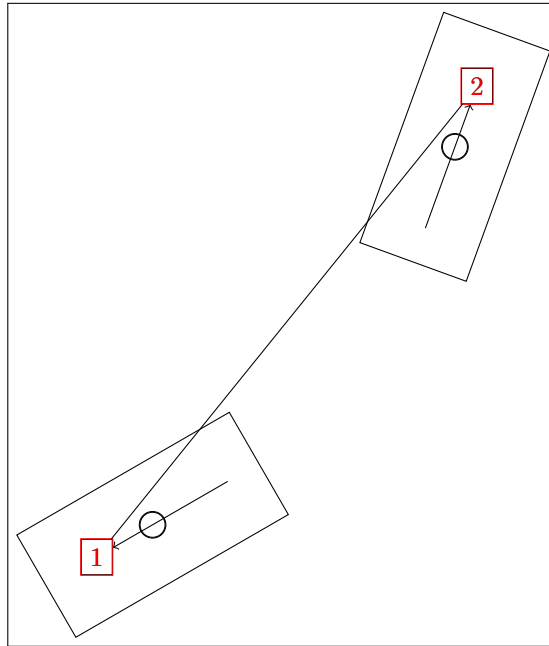
\node[red] at (pic-1-1) {1};
\node[red] at (pic-2-2) {2};

\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 `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).

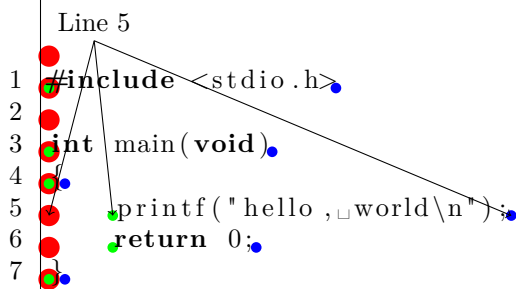
The line numbers *should* match up with the line numbers in the code in that any initial offset is also applied.

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}
```



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, 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 L^AT_EX3.

```

1 \RequirePackage{expl3, l3keys2e, xparse}
2 \tikzset{%
3   remember picture with id/.style={%
4     remember picture,
5     overlay,
6     save picture id=#1,
7   },

```

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.

```

8   every picture/.append style={%

```



```

9     execute at end picture={%
10     \ifpgfrememberpicturerepositiononpage%
11     \edef\pgf@temp{%
12     \noexpand\write\noexpand\pgfutil@auxout{%
13     \string\savepicturepage%
14     {\pgfpictureid}{\noexpand\arabic{page}}}%
15     }%
16     }%
17     \pgf@temp
18     \fi%
19 },
20 },

```

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

```

21 save picture id/.code={%
22     \protected@write\pgfutil@auxout{}{%
23     \string\savepointas%
24     {\tikzmark@pp@name{#1}}{\pgfpictureid}{0pt}{0pt}}%
25 },

```

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

```

26 if picture id/.code args={#1#2#3}{%
27     \@ifundefined{save@pt@\tikzmark@pp@name{#1}}{%
28     \pgfkeysalso{#3}%
29     }{
30     \pgfkeysalso{#2}%
31     }
32 },

```

Page handling

```

33 next page/.is choice,
34 next page vector/.initial={\pgfqpoint{0pt}{0pt}},
35 next page/below/.style={%
36     next page vector={\pgfqpoint{0pt}{-\the\paperheight}}%
37 },
38 next page/above/.style={%
39     next page vector={\pgfqpoint{0pt}{\the\paperheight}}%
40 },
41 next page/left/.style={%
42     next page vector={\pgfqpoint{-\the\paperwidth}{0pt}}%
43 },
44 next page/right/.style={%
45     next page vector={\pgfqpoint{\the\paperwidth}{0pt}}%
46 },
47 next page/ignore/.style={%
48     next page vector={\pgfqpoint{0pt}{0pt}}%
49 },
50 if tikzmark on current page/.code n args={3}{%
51     \@ifundefined{save@pt@\tikzmark@pp@name{#1}}{%

```

```

52     \pgfkeysalso{#3}%
53 }{%
54     \ifundefined{save@pg@\csname save@pt@tikzmark@pp@name{#1}\endcsname}{%
55         \pgfkeysalso{#3}%
56     }{%
57         \ifnum\csname save@pg@\csname save@pt@tikzmark@pp@name{#1}\endcsname\endcsname=\the\va
58         \pgfkeysalso{#2}%
59         \else
60         \pgfkeysalso{#3}%
61         \fi
62     }%
63 }%
64 },
65 if tikzmark on page/.code n args={4}{%
66     \ifundefined{save@pt@tikzmark@pp@name{#1}}{%
67         \pgfkeysalso{#4}%
68     }{%
69         \ifundefined{save@pg@\csname save@pt@tikzmark@pp@name{#1}@label\endcsname}{%
70             \pgfkeysalso{#4}%
71         }{%
72             \ifnum\csname save@pg@\csname save@pt@tikzmark@pp@name{#1}\endcsname\endcsname=#2\rela
73             \pgfkeysalso{#3}%
74             \else
75             \pgfkeysalso{#4}%
76             \fi
77         }%
78     }%
79 },

```

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

```

80 tikzmark prefix/.initial=,%
81 tikzmark suffix/.initial=,%
82 }

```

\tikzmark@pp@name

```

83 \def\tikzmark@pp@name#1{%
84     \csname pgfk@tikz/tikzmark prefix\endcsname%
85     #1%
86     \csname pgfk@tikz/tikzmark suffix\endcsname%
87 }%

```

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

```

88 \def\savepointas#1#2#3#4{%
89     \expandafter\gdef\csname save@pt@#1\endcsname{#2}%
90     \expandafter\gdef\csname save@pt@#1@offset\endcsname%
91     {\pgfqpoint{#3}{#4}}%
92 }
93 \def\savepicturepage#1#2{%
94     \expandafter\gdef\csname save@pg@#1\endcsname{#2}%

```

```

95 }

\tikzmarkalias  Alias a tikzmark to another name (used in tikzmarknode)
96 \def\tikzmarkalias#1#2{%
97   \pgf@node@gnamelet{save@pt@#1}{save@pt@#2}%
98   \pgf@node@gnamelet{save@pt@#1@offset}{save@pt@#2@offset}%
99 }

\tmk@labeldef  Auxiliary command for the coordinate system.
100 \def\tmk@labeldef#1,#2\@nil{%
101   \edef\tmk@label{\tikzmark@pp@name{#1}}%
102   \def\tmk@def{#2}%
103 }

pic  This defines the new coordinate system.
104 \tikzdeclarecoordinatesystem{pic}{%
105   \pgfutil@in@,{#1}%
106   \ifpgfutil@in@%
107     \tmk@labeldef#1\@nil
108   \else
109     \tmk@labeldef#1,(0pt,0pt)\@nil
110   \fi
111   \@ifundefined{save@pt@\tmk@label}{%
112     \tikz@scan@one@point\pgfutil@firstofone\tmk@def
113   }{%
114     \pgfsys@getposition{\csname save@pt@\tmk@label\endcsname}%
115     \save@orig@pic%
116     \pgfsys@getposition{\pgfpictureid}\save@this@pic%
117     \pgf@process{\pgfp@point@origin\save@this@pic}%
118     \pgf@xa=\pgf@x
119     \pgf@ya=\pgf@y
120     \pgf@process{\pgfp@point@origin\save@orig@pic}%
121     \advance\pgf@x by -\pgf@xa
122     \advance\pgf@y by -\pgf@ya
123     \pgf@xa=\pgf@x
124     \pgf@ya=\pgf@y
125     \pgf@process%
126     {\pgfp@point@origin\csname save@pt@\tmk@label @offset\endcsname}%
127     \advance\pgf@xa by \pgf@x
128     \advance\pgf@ya by \pgf@y
129     \@ifundefined{save@pg@\csname save@pt@\tmk@label\endcsname}{%}{%
130       \@ifundefined{save@pg@\pgfpictureid}{%}{%
131         \pgfkeysvalueof{/tikz/next page vector}%
132         \edef\tmk@pg{\the\numexpr \csname save@pg@\csname save@pt@\tmk@label\endcsname\endcsname
133         \advance \pgf@xa by %
134         \tmk@pg\pgf@x\relax
135         \advance \pgf@ya by %
136         \tmk@pg\pgf@y\relax
137       }%

```

```

138     }%
139     \pgf@x=\pgf@xa
140     \pgf@y=\pgf@ya
141     \pgftransforminvert
142     \pgf@pos@transform{\pgf@x}{\pgf@y}%
143 }%
144 }

\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
145 \newcommand\tikzmark@outside[2] [] {%
146 \tikz[remember picture with id=#2]{#1}%
147 }

This is for when we're inside a tikzpicture environment
148 \def\tikzmark@inside#1#2{%
149 \tikzset{remember picture}%
150 \tikz@scan@one@point\pgfutil@firstofone#2\relax
151 \protected@write\pgfutil@auxout{}{%
152 \string\savepointas{\tikzmark@pp@name{#1}}{\pgfpictureid}{\the\pgf@x}{\the\pgf@y}}%
153 }

And finally, the ultimate invoker:
154 \def\tikzmark{%
155 \ifx\pgfpictureid\@undefined
156 \let\tikzmark@next=\tikzmark@outside
157 \else
158 \relax
159 \ifx\scope\tikz@origscope\relax
160 \let\tikzmark@next=\tikzmark@outside
161 \else
162 \let\tikzmark@next=\tikzmark@inside
163 \fi
164 \fi
165 \tikzmark@next%
166 }

\pgfmark
167 \newcommand\pgfmark[1]{%
168 \bgroup
169 \global\advance\pgf@picture@serial@count by1\relax%
170 \edef\pgfpictureid{\pgfid\the\pgf@picture@serial@count}%
171 \pgfsys@markposition{\pgfpictureid}%
172 \edef\pgf@temp{%
173 \noexpand\write\noexpand\pgfutil@auxout{%
174 \string\savepicturepage{\pgfpictureid}{\noexpand\arabic{page}}}}}%
175 \pgf@temp

```

```

176 \protected@write\pgfutil@auxout{}\{%
177 \string\savepointas\tikzmark@pp@name{#1}}\pgfpictureid}{Opt}{Opt}}%
178 \egroup
179 }

```

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

`\tikzmark<>`

```

180 \@ifclassloaded{beamer}{
181 \renewcommand<>\tikzmark@outside}[2][\%
182 \only#3{\beameroriginal\tikzmark@outside}[{#1}]{#2}}%
183 }
184 \renewcommand<>\tikzmark@inside}[2][\%
185 \only#3{\beameroriginal\tikzmark@inside}[{#1}]{#2}}%
186 }
187 }{}

```

`\pgfmark<>`

```

188 \@ifclassloaded{beamer}{
189 \renewcommand<>\pgfmark}[1]{\only#2{\beameroriginal\pgfmark}[{#1}]}
190 }{}

```

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

```

191 \@ifclassloaded{beamer}{
192 \tikzset{
193 tikzmark suffix=-\the\beamer@slideinframe
194 }
195 }{}

```

`\iftikzmark`

```

196 \newif\iftikzmark@
197 \newcommand\iftikzmark[3]{\%
198 \ifundefined{save@pt\tikzmark@pp@name{#1}}{\%
199 #3%
200 }{\%
201 #2%
202 }%
203 }%

```

A version suitable for `\if ... \else ... \fi`.

```

204 \newcommand\iftikzmarkexists[1]{\%
205 \ifundefined{save@pt\tikzmark@pp@name{#1}}{\%
206 \tikzmark@false%
207 }{\%
208 \tikzmark@true%
209 }%
210 \iftikzmark@
211 }%

```

\iftikzmarkonpage

```

212 \newcommand\iftikzmarkonpage[2]{%
213   \@ifundefined{save@pt@\tikzmark@pp@name{#1}}{%
214     \tikzmark@false
215   }{%
216     \@ifundefined{save@pg@\csname save@pt@\tikzmark@pp@name{#1}\endcsname}{%
217       \tikzmark@false
218     }{%
219       \ifnum\csname save@pg@\csname save@pt@\tikzmark@pp@name{#1}\endcsname\endcsname=#2\relax%
220       \tikzmark@true
221     }else
222       \tikzmark@false
223     \fi
224   }%
225 }%
226 \iftikzmark@
227 }

```

\iftikzmarkoncurrentpage

```

228 \newcommand\iftikzmarkoncurrentpage[1]{%
229   \@ifundefined{save@pt@\tikzmark@pp@name{#1}}{%
230     \tikzmark@false
231   }{%
232     \@ifundefined{save@pg@\csname save@pt@\tikzmark@pp@name{#1}\endcsname}{%
233       \tikzmark@false
234     }{%
235       \ifnum\csname save@pg@\csname save@pt@\tikzmark@pp@name{#1}\endcsname\endcsname=\the\valu
236       \tikzmark@true
237     }else
238       \tikzmark@false
239     \fi
240   }%
241 }%
242 \iftikzmark@
243 }

```

\subnode Note: much of this code was inevitably adapted from the node defining code in the TikZ/PGF sources.

```

244 \def\subnode@#1#2#3{%
245   \begingroup
246   \pgfmark{#2}%
247   \setbox\pgfnodeparttextbox=\hbox\bgroup #3\egroup
248   \def\tikz@fig@name{#2}%
249   \tikzset{every subnode/.try,#1}%
250   \pgfpointorigin
251   \tikz@scan@one@point\pgfutil@firstofone(pic cs:#2)\relax
252   \advance\pgf@x by .5\wd\pgfnodeparttextbox
253   \advance\pgf@y by .5\ht\pgfnodeparttextbox
254   \advance\pgf@y by -.5\dp\pgfnodeparttextbox

```

```

255 \pgftransformshift{}%
256 \setbox\@tempboxa=\hbox\bgroup
257 {%
258 \let\pgf@sh@savemacros=\pgfutil@empty% MW
259 \let\pgf@sh@savedpoints=\pgfutil@empty%
260 \def\pgf@sm@shape@name{rectangle}% CJ % TT added prefix!
261 \pgf@sh@s@rectangle%
262 \pgf@sh@savedpoints%
263 \pgf@sh@savemacros% MW
264 \pgftransformshift{%
265 \pgf@sh@reanchor{rectangle}{center}%
266 \pgf@x=-\pgf@x%
267 \pgf@y=-\pgf@y%
268 }%
269 \expandafter\pgfsavepgf@process\csname pgf@sh@sa@\tikz@fig@name\endcsname{%
270 \pgf@sh@reanchor{rectangle}{center}% FIXME : this is double work!
271 }%
272 % Save the saved points and the transformation matrix
273 \edef\pgf@node@name{\tikz@fig@name}%
274 \ifx\pgf@node@name\pgfutil@empty%
275 \else%
276 \expandafter\xdef\csname pgf@sh@ns@\pgf@node@name\endcsname{rectangle}%
277 \edef\pgf@sh@temp{%
278 \noexpand\gdef\expandafter\noexpand\csname pgf@sh@np@\pgf@node@name\endcsname}%
279 \expandafter\pgf@sh@temp\expandafter{%
280 \pgf@sh@savedpoints}%
281 \edef\pgf@sh@temp{%
282 \noexpand\gdef\expandafter\noexpand\csname pgf@sh@ma@\pgf@node@name\endcsname}% MW
283 \expandafter\pgf@sh@temp\expandafter{\pgf@sh@savemacros}% MW
284 \pgfgettransform\pgf@temp
285 \expandafter\xdef\csname pgf@sh@nt@\pgf@node@name\endcsname{\pgf@temp}%
286 \expandafter\xdef\csname pgf@sh@pi@\pgf@node@name\endcsname{\pgfpictureid}%
287 \fi%
288 }%
289 \egroup
290 \box\pgfnodeparttextbox
291 \endgroup
292 }
293
294 \newcommand\subnode[3][]{%
295 \ifmmode
296 \mathchoice{%
297 \subnode@{#1}{#2-d}{\(\displaystyle #3\)}%
298 }{%
299 \subnode@{#1}{#2-t}{\(\textstyle #3\)}%
300 }{%
301 \subnode@{#1}{#2-s}{\(\scriptstyle #3\)}%
302 }{%
303 \subnode@{#1}{#2-ss}{\(\scriptscriptstyle #3\)}%
304 }%

```

```

305 \let\pgf@nodecallback\pgfutil@gobble
306 \def\tzmk@prfx{pgf@sys@pdf@mark@pos@pgfid}%
307 \edef\tzmk@pic{\tzmk@prfx\the\pgf@picture@serial@count}
308 \expandafter\ifx\csname\tzmk@pic\endcsname\relax
309 \edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-1\relax}%
310 \expandafter\ifx\csname\tzmk@pic\endcsname\relax
311 \edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-2\relax}%
312 \expandafter\ifx\csname\tzmk@pic\endcsname\relax
313 \edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-3\relax}%
314 \expandafter\ifx\csname\tzmk@pic\endcsname\relax
315 % \pgfnodealias{#2}{#2-t}% %% Fallback
316 \else
317 \pgfnodealias{#2}{#2-d}%
318 \tikzmarkalias{#2}{#2-d}%
319 \fi
320 \else
321 \pgfnodealias{#2}{#2-t}%
322 \tikzmarkalias{#2}{#2-t}%
323 \fi
324 \else
325 \pgfnodealias{#2}{#2-s}%
326 \tikzmarkalias{#2}{#2-s}%
327 \fi
328 \else
329 \pgfnodealias{#2}{#2-ss}%
330 \tikzmarkalias{#2}{#2-ss}%
331 \fi
332 \else
333 \subnode@{#1}{#2}{#3}%
334 \fi
335 }
336

```

`\tikzmarknode` The `\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 `\mathchoice`: the `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.

```

337 \def\tikzmarknode@#1#2#3{%
338 \tikz[%
339     remember picture,
340     baseline=(#2.base),
341     save picture id={#2},
342     every tikzmarknode picture/.try
343 ] {
344     \node[

```



```

345     anchor=base,
346     inner sep=0pt,
347     minimum width=0pt,
348     name={#2},
349     node contents={#3},
350     every tikzmarknode/.try,
351     #1
352   ]}%
353 }
354
355 \newcommand\tikzmarknode[3] [] {%
356   \ifmmode
357     \mathchoice{%
358       \tikzmarknode@{#1}{#2-d}{\(\displaystyle #3\)}%
359     }{%
360       \tikzmarknode@{#1}{#2-t}{\(\textstyle #3\)}%
361     }{%
362       \tikzmarknode@{#1}{#2-s}{\(\scriptstyle #3\)}%
363     }{%
364       \tikzmarknode@{#1}{#2-ss}{\(\scriptscriptstyle #3\)}%
365     }%
366   \let\pgf@nodecallback\pgfutil@gobble
367   \def\tzmk@prfx{\pgf@sys@pdf@mark@pos@pgfid}%
368   \edef\tzmk@pic{\tzmk@prfx\the\pgf@picture@serial@count}%
369   \expandafter\ifx\csname\tzmk@pic\endcsname\relax
370     \edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-1\relax}%
371     \expandafter\ifx\csname\tzmk@pic\endcsname\relax
372       \edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-2\relax}%
373       \expandafter\ifx\csname\tzmk@pic\endcsname\relax
374         \edef\tzmk@pic{\tzmk@prfx\the\numexpr\the\pgf@picture@serial@count-3\relax}%
375         \expandafter\ifx\csname\tzmk@pic\endcsname\relax
376           % No fallback due to potential nested mathchoices
377           \else
378             \pgfnodealias{#2}{#2-d}%
379             \tikzmarkalias{#2}{#2-d}%
380           \fi
381           \else
382             \pgfnodealias{#2}{#2-t}%
383             \tikzmarkalias{#2}{#2-t}%
384           \fi
385           \else
386             \pgfnodealias{#2}{#2-s}%
387             \tikzmarkalias{#2}{#2-s}%
388           \fi
389           \else
390             \pgfnodealias{#2}{#2-ss}%
391             \tikzmarkalias{#2}{#2-ss}%
392           \fi
393           \else
394             \tikzmarknode@{#1}{#2}{#3}%

```

```

395 \fi
396 }

\usetikzmarklibrary
397 \def\usetikzmarklibrary{%
398 \pgfutil@ifnextchar[{\use@tikzmarklibrary}{\use@tikzmarklibrary}%
399 }%}
400 \def\use@tikzmarklibrary[#1]{\use@tikzmarklibrary{#1}}
401 \def\use@tikzmarklibrary#1{%
402 \edef\pgf@list{#1}%
403 \pgfutil@for\pgf@temp:=\pgf@list\do{%
404 \expandafter\pgfkeys@spdef\expandafter\pgf@temp\expandafter{\pgf@temp}%
405 \ifx\pgf@temp\pgfutil@empty
406 \else
407 \expandafter\ifx\csname tikzmark@library@\pgf@temp @loaded\endcsname\relax%
408 \expandafter\global\expandafter\let%
409 \csname tikzmark@library@\pgf@temp @loaded\endcsname=\pgfutil@empty%
410 \expandafter\edef\csname tikzmark@library@#1@atcode\endcsname{\the\catcode'\@}
411 \expandafter\edef\csname tikzmark@library@#1@barcode\endcsname{\the\catcode'\|}
412 \catcode'\@=11
413 \catcode'\|=12
414 \pgfutil@InputIfFileExists{tikzmarklibrary\pgf@temp.code.tex}{%{
415 \PackageError{tikzmark}{I did not find the tikzmark extras library '\pgf@temp'.}{%}
416 }%
417 \catcode'\@=\csname tikzmark@library@#1@atcode\endcsname
418 \catcode'\|=\csname tikzmark@library@#1@barcode\endcsname
419 \fi%
420 \fi
421 }%
422 }
423

```

The `save` node code is written in L^AT_EX3.

```
424 \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.

```
425 \prop_new:N \g__sn_prop
```

We’ll need a couple of spare token lists

```
426 \tl_new:N \l__sn_tmpa_tl
```

```
427 \tl_new:N \l__sn_tmpb_tl
```

Another useful token list

```
428 \tl_new:N \l__open_bracket_tl
```

```
429 \tl_set:Nn \l__open_bracket_tl {[} %]
```

This token list is used for our current node group name

```
430 \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.

```
431 \clist_new:N \l__sn_nodes_clist
```

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

```
432 \bool_new:N \l__sn_file_bool
```

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

```
433 \bool_new:N \g__sn_preamble_bool
```

```
434 \bool_set_true:N \g__sn_preamble_bool
```

```
435 \msg_new:nnn {tikzmark} {no file} {File~ "#1"~ doesn't~ exist.}
```

```
436 \msg_new:nnn {tikzmark} {loading nodes} {Loading~ nodes~ from~ "#1".}
```

Dimensions and token lists for shifting

```
437 \dim_new:N \l__sn_x_dim
```

```
438 \dim_new:N \l__sn_y_dim
```

```
439 \dim_new:N \l__sn_xa_dim
```

```
440 \dim_new:N \l__sn_ya_dim
```

```
441 \tl_new:N \l__sn_centre_tl
```

```
442
```

```
443 \tl_new:N \l__sn_transformation_tl
```

```
444 \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

```
445 \iow_new:N \g__sn_stream
```

```
446 \bool_new:N \g__sn_stream_bool
```

```
447 \tl_new:N \g__sn_filename_tl
```

```
448 \tl_gset:Nx \g__sn_filename_tl {\c_sys_jobname_str}
```

```
449
```

```
450 \cs_new_nopar:Npn \sn_open_stream:
```

```
451 {
```

```
452   \bool_if:NF \g__sn_stream_bool
```

```
453   {
```

```
454     \iow_open:Nn \g__sn_stream {\tl_use:N \g__sn_filename_tl .nodes}
```

```
455     \bool_gset_true:N \g__sn_stream_bool
```

```
456   }
```

```
457 }
```

```
458
```

```
459 \AtEndDocument
```

```
460 {
```

```
461   \ExplSyntaxOn
```

```
462   \bool_if:NT \g__sn_stream_bool
```

```
463   {
```

```
464     \iow_close:N \g__sn_stream
```

```
465   }
```

```
466   \ExplSyntaxOff
```

```
467 }
```

LaTeX3 wrappers around some PGF functions (to avoid @-catcode issues)

```
468 \makeatletter
```

```
469 \cs_set_eq:NN \tikz_set_node_name:n \tikz@pp@name
```

```
470 \cs_set_eq:NN \tikz_fig_must_be_named: \tikz@fig@mustbenamed
```

```

471
472 \cs_new_nopar:Npn \tikz_scan_point:n #1
473 {
474   \tikz@scan@one@point\pgfutil@firstofone#1\relax
475 }
476
477 \cs_new_nopar:Npn \tikz_scan_point:NNn #1#2#3
478 {
479   \tikz@scan@one@point\pgfutil@firstofone#3\relax
480   \dim_set_eq:NN #1 \pgf@x
481   \dim_set_eq:NN #2 \pgf@y
482 }
483
484 \makeatother
485 \cs_generate_variant:Nn \tikz_scan_point:n {V}
486 \cs_generate_variant:Nn \tikz_scan_point:NNn {NNV}

\save_nodes:Nn 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.

487 \cs_new_nopar:Npn \save_nodes:Nn #1#2
488 {
  Clear our token lists
489   \tl_clear:N \l__sn_tmpa_tl
  Set the centre of the picture
490   \tikz_scan_point:NNn \l__sn_x_dim \l__sn_y_dim {(current~ bounding~ box.center)}
491   \dim_set:Nn \l__sn_x_dim {-\l__sn_x_dim}
492   \dim_set:Nn \l__sn_y_dim {-\l__sn_y_dim}
493   \tl_set:Nx \l__sn_centre_tl {
494     {1}{0}{0}{1}{\dim_use:N \l__sn_x_dim}{\dim_use:N \l__sn_y_dim}
495   }
  Iterate over the list of node names
496   \clist_map_inline:nn {#2}
497   {
    Test to see if the node has been defined
498     \tl_if_exist:cT {pgf@sh@ns@##1}
499     {
      The node information is stored in a series of macros of the form \pgf@sh@XX@nodename
      where XX is one of the following.
500       \clist_map_inline:nn {ns,np,ma,pi}
501       {
        Our token list will look like:
        \tl_set:cn {pgf@sh@XX@nodename} <current contents of that macro>
        This will restore \pgf@sh@XX@nodename to its current value when this list is
        invoked.

```

This part puts the `\tl_set:cn {pgf@sh@XX@nodename}` in place

```

502     \tl_put_right:Nn \l__sn_tmpa_tl
503     {
504         \tl_gset:cn {pgf@sh@####1@ \tikz_set_node_name:n{##1} }
505     }

```

Now we put the current contents in place. We're doing this in an expansive context to get at the contents. The `\exp_not:v` part takes the current value of `\pgf@sh@XX@nodename` and puts it in place, preventing further expansion.

```

506     \tl_if_exist:cTF {pgf@sh@####1@##1}
507     {
508         \tl_put_right:Nx \l__sn_tmpa_tl {
509             {\exp_not:v {pgf@sh@####1@ \tikz_set_node_name:n {##1}}}
510         }
511     }
512     {
513         \tl_put_right:Nx \l__sn_tmpa_tl {{{
514     }
515 }
516 \tl_put_right:Nn \l__sn_tmpa_tl
517 {
518     \tl_gset:cn {pgf@sh@nt@ \tikz_set_node_name:n{##1} }
519 }
520 \compose_transformations:NVv \l__sn_tmpb_tl \l__sn_centre_tl {pgf@sh@nt@##1}
521 \tl_put_right:Nx \l__sn_tmpa_tl {{{\exp_not:V \l__sn_tmpb_tl}}}
522 \tl_put_right:Nn \l__sn_tmpa_tl {
523     \transform_node:Nn \l__sn_transformation_tl {
524         \tikz_set_node_name:n{##1}
525     }
526 }
527 }
528 }

```

Once we've assembled our token list, we store it in the given token list

```

529 \tl_set_eq:NN #1 \l__sn_tmpa_tl
530 }

```

`\save_nodes_to_list:nn` Save the nodes to a list, given a key

```

531 \cs_new_nopar:Npn \save_nodes_to_list:nn #1#2
532 {
533     \save_nodes:Nn \l__sn_tmpa_tl {#2}
534     \prop_gput:NnV \g__sn_prop {#1} \l__sn_tmpa_tl
535 }

```

`\save_nodes_to_file:n` Save the nodes to a file

```

536 \cs_new_nopar:Npn \save_nodes_to_file:n #1
537 {
538     \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

```

539 \sn_open_stream:
540 \iow_now:Nx \g__sn_stream
541 {
542   \iow_newline:
543   \exp_not:V \l__sn_tmpa_tl
544   \iow_newline:
545 }
546 }

547 \cs_generate_variant:Nn \save_nodes_to_list:nn {VV}
548 \cs_generate_variant:Nn \save_nodes_to_file:n {V}

```

\restore_nodes_from_list:n

```

549 \cs_new_nopar:Npn \restore_nodes_from_list:n #1
550 {
  Restoring nodes is simple: look in the property list for the key and if it exists,
  invoke the macro stored there.
551   \prop_get:NnNT \g__sn_prop {#1} \l__sn_tmpa_tl
552   {
553     \tl_use:N \l__sn_tmpa_tl
554   }
555 }

```

\restore_nodes_from_file:n

```

556 \cs_new_nopar:Npn \restore_nodes_from_file:n #1
557 {
558   \file_if_exist:nTF {#1.nodes}
559   {
560     \msg_log:nnn {tikzmark} {loading nodes} {#1}
561     \ExplSyntaxOn
562     \file_input:n {#1.nodes}
563     \ExplSyntaxOff
564   }
565   {
566     \msg_warning:nnn {tikzmark} {no file} {#1}
567   }
568 }
569 \AtBeginDocument{\bool_gset_false:N \g__sn_preamble_bool}

```

\compose_transformations:Nnn Compose PGF transformations #2 * #3, storing the result in #1

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:

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

```

570 \cs_new_nopar:Npn \compose_transformations:Nnn #1#2#3
571 {

```

```

572 \tl_gset:Nx #1
573 {
574   {\fp_eval:n {
575     \tl_item:nn {#2} {1}
576     * \tl_item:nn {#3} {1}
577     +
578     \tl_item:nn {#2} {3}
579     * \tl_item:nn {#3} {2}
580   }
581 }
582 {\fp_eval:n {
583   \tl_item:nn {#2} {2}
584   * \tl_item:nn {#3} {1}
585   +
586   \tl_item:nn {#2} {4}
587   * \tl_item:nn {#3} {2}
588 }
589 }
590 {\fp_eval:n {
591   \tl_item:nn {#2} {1}
592   * \tl_item:nn {#3} {3}
593   +
594   \tl_item:nn {#2} {3}
595   * \tl_item:nn {#3} {4}
596 }
597 }
598 {\fp_eval:n {
599   \tl_item:nn {#2} {2}
600   * \tl_item:nn {#3} {3}
601   +
602   \tl_item:nn {#2} {4}
603   * \tl_item:nn {#3} {4}
604 }
605 }
606 {\fp_to_dim:n {
607   \tl_item:nn {#2} {1}
608   * \tl_item:nn {#3} {5}
609   +
610   \tl_item:nn {#2} {3}
611   * \tl_item:nn {#3} {6}
612   +
613   \tl_item:nn {#2} {5}
614 }
615 }
616 {\fp_to_dim:n {
617   \tl_item:nn {#2} {2}
618   * \tl_item:nn {#3} {5}
619   +
620   \tl_item:nn {#2} {4}
621   * \tl_item:nn {#3} {6}

```

```

622         +
623         \tl_item:nn {#2} {6}
624     }
625 }
626 }
627 }

```

```

628 \cs_generate_variant:Nn \compose_transformations:Nnn {cVv,NVv,NVn,NvV,NnV}

```

```

\tl_set_transform_node:Nn

```

```

629 \cs_new_nopar:Npn \transform_node:Nn #1#2
630 {
631     \compose_transformations:cVv {pgf@sh@nt@#2} #1 {pgf@sh@nt@#2}
632 }

```

```

\set_transform_from_node:n

```

```

633 \cs_new_nopar:Npn \set_transform_from_node:n #1
634 {
635     \tl_set_eq:Nc \l__sn_transformation_tl {pgf@sh@nt@#1}
636     \tikz_scan_point:NNn \l__sn_x_dim \l__sn_y_dim {(#1.center)}
637
638     \dim_set:Nn \l__sn_x_dim {\l__sn_x_dim - \tl_item:cn {pgf@sh@nt@#1}{5}}
639     \dim_set:Nn \l__sn_y_dim {\l__sn_y_dim - \tl_item:cn {pgf@sh@nt@#1}{6}}
640
641     \compose_transformations:NnV \l__sn_transformation_tl {
642         {1}{0}{0}{1}{\dim_use:N \l__sn_x_dim}{\dim_use:N \l__sn_y_dim}
643     } \l__sn_transformation_tl
644 }

```

```

645 \cs_generate_variant:Nn \set_transform_from_node:n {v}

```

Set the TikZ keys for access to the above commands.

```

646 \tikzset{
647     set~ saved~ nodes~ file~ name/.code={
648         \tl_gset:Nx \g__sn_filename_tl {#1}
649     },
650     transform~ saved~ nodes/.code={
651         \set_transform_from_node:v {tikz@last@fig@name}
652     },
653     set~ node~ group/.code={
654         \tl_set:Nn \l__sn_group_tl {#1}
655         \pgfkeysalso{
656             execute~ at~ end~ scope={
657                 \maybe_save_nodes:
658             }
659         }
660     },

```

Are we saving to a file?

```

661     save~ nodes~ to~ file/.code={
662         \tl_if_eq:nnTF {#1}{false}

```



```

663 {
664   \bool_set_false:N \l__sn_file_bool
665 }
666 {
667   \bool_set_true:N \l__sn_file_bool
668 }
669 \pgfkeysalso{
670   execute~ at~ end~ scope={
671     \maybe_save_nodes:
672   }
673 }
674 },

Append current node to the list of nodes to be saved
675 save~ node/.code={
676   \tikz_fig_must_be_named:
677   \pgfkeysalso{append~ after~ command={
678     \pgfextra{
679       \clist_gput_right:Nv \l__sn_nodes_clist {tikz@last@fig@name}
680     }
681   }
682 }
683 },

Restore nodes from file
684 restore~ nodes~ from~ file/.code={
685   \bool_if:NTF \g__sn_preamble_bool
686   {
687     \restore_nodes_from_file:n {#1}
688   }
689   {
690     \tikz_fig_must_be_named:
691     \pgfkeysalso{append~ after~ command={
692       \pgfextra{
693         \scope
694         \split_argument:NNn \tikzset \restore_nodes_from_file:n {#1}
695         \endscope
696       }
697     }
698   }
699 }
700 },

701 restore~ nodes~ from~ file/.default = \g__sn_filename_tl,

Restore nodes from list
702 restore~ nodes~ from~ list/.code={
703   \tikz_fig_must_be_named:
704   \pgfkeysalso{append~ after~ command={
705     \pgfextra{
706       \scope
707       \split_argument:NNn \tikzset \restore_nodes_from_list:n {#1}

```

```

708         \endscope
709     }
710 }
711 }
712 }
713 }
714 \cs_generate_variant:Nn \clist_gput_right:Nn {Nv}

\split_argument:NNn
715 \cs_new_nopar:Npn \split_argument:NNn #1#2#3
716 {
717     \tl_set:Nx \l__sn_tmpa_tl {\tl_head:n {#3}}
718     \tl_if_eq:NNTF \l__sn_tmpa_tl \l__open_bracket_tl
719     {
720         \split_argument_aux:NNp #1#2#3
721     }
722     {
723         #2 {#3}
724     }
725 }

\split_argument_aux:NNp
726 \cs_new_nopar:Npn \split_argument_aux:NNp #1#2[#3]#4
727 {
728     #1 {#3}
729     #2 {#4}
730 }

\maybe_save_nodes:
731 \cs_new_nopar:Npn \maybe_save_nodes:
732 {
733     \clist_if_empty:NF \l__sn_nodes_clist
734     {
735         \bool_if:NTF \l__sn_file_bool
736         {
737             \save_nodes_to_file:V \l__sn_nodes_clist
738         }
739         {
740             \tl_if_empty:NF \l__sn_group_tl
741             {
742                 \save_nodes_to_list:VV \l__sn_group_tl \l__sn_nodes_clist
743             }
744         }
745         \clist_gclear:N \l__sn_nodes_clist
746     }
747 }

748 \ExplSyntaxOff

```

7.2 Listings

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

```

749 \@ifpackageloaded{listings}{%

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

EveryLine This hook places the mark at the start of the line.
751 \lst@AddToHook{EveryLine}{%
752     \begingroup
753     \advance\c@lstnumber by 1\relax
754     \pgfmark{line-\lst@name-\the\c@lstnumber-start}%
755     \endgroup
756 }

EOL This hook places the mark at the end of the line and resets the conditional for
    placing the first mark.
757 \lst@AddToHook{EOL}{\pgfmark{line-\lst@name-\the\c@lstnumber-end}%
758 \global\lst@linemarktrue
759 }

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.
760 \lst@AddToHook{OutputBox}{%
761     \iflst@linemark
762     \pgfmark{line-\lst@name-\the\c@lstnumber-first}%
763     \global\lst@linemarkfalse
764     \fi
765 }

\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.
766 \def\tikzmk@lst@fnum#1\relax#2\@STOP{%
767     \def\@test{#2}%
768     \ifx\@test\@empty
769     \def\tikzmk@lst@start{0}%
770     \else
771     \@tempcnta=#1\relax
772     \advance\@tempcnta by -1\relax
773     \def\tikzmk@lst@start{\the\@tempcnta}%
774     \fi
775 }

Init Adds a mark at the start of the listings environment.
776 \lst@AddToHook{Init}{%
777     \expandafter\tikzmk@lst@fnum\lst@firstnumber\relax\@STOP
778     \pgfmark{line-\lst@name-\tikzmk@lst@start-start}%
779 }
```

```
780 }{%  
781     \PackageError{tikzmark listings}{The listings package has not been loaded.}{}  
782 }
```