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 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 \texttt{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 \texttt{tikzmark} macro in 2009 for use in lecture slides prepared with the \texttt{beamer} package. Its original definition was:

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

Its first use was in the (inelegant) code:

\begin{verbatim}
\begin{frame}

\begin{tikzpicture}[overlay, remember picture]
\useasboundingbox (0,0);
\draw<2-|trans: 0|handout: 0>[red,->] (bsp) .. controls +(-1,-1) and
($\text{cnvs.north}+(1,1)$) .. ($\text{cnvs.north}+(0,1)$) .. controls
($\text{cnvs.north}+(-1,1)$) 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
($\text{nvs.north}+(0,1.5)$) .. ($\text{nvs.north}+(0,1.5)$) .. controls
($\text{nvs.north}+(-1.5,1.5)$) and +(-1.5,0) .. (nvs.north);
\draw<5-|trans: 0|handout: 0>[purple,->] (vector) .. controls +(-1,-1) and
($\text{vsp.north}+(2,2)$) .. ($\text{vsp.north}+(0,2)$) .. controls
($\text{vsp.north}+(-2,2)$) and +(-2,0) .. (vsp.north);
\end{tikzpicture}

\begin{theorem}
\centering
\big($C([0,1],\mathbb{R}),d_\infty\big)$
\end{theorem}
\pause
\bigskip

\begin{itemize}
\item \textcolor<.(2)->{green!50!black}Complete
\item \textcolor<.(3)->{blue}Normed
\item \textcolor<.(4)->{purple}Vector
\end{itemize}
\pause
\bigskip
\bigskip
\end{frame}
\end{verbatim}
Structure of Continuous Functions

Theorem

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

is a

\textit{Banach space}

\begin{itemize}
\item Complete normed vector space.
\item Cauchy sequences converge.
\item Metric from a norm.
\item Functions behave like vectors.
\end{itemize}

\begin{frame}
\begin{itemize}[<+->]
\item \tikzmark{mcplt} \textcolor{green!50!black}{Cauchy sequences converge.}
\item \tikzmark{nvs} \textcolor{blue}{Metric from a norm.}
\item \tikzmark{vsp} \textcolor{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 \ref{answer}, 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.

4.1 Core Commands

1. \tikzmark\{\textit{drawing command}\}\{\textit{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 \ldots ; 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 \texttt{beamer} class is loaded then this command is made overlay-aware.

2. \tikzmark\{\textit{name}\}\{\textit{coordinate}\}

v1.2 of the \tikzmark package introduced a new variant of \tikzmark which works inside a \texttt{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 \texttt{tikzpicture} and so it is now possible to use \tikzmark inside a \texttt{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 \texttt{\else} and \texttt{\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 \texttt{\else} and \texttt{\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 \texttt{\else} and \texttt{\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 got 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. \texttt{(pic cs:<name>)} or \texttt{(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. \texttt{/tikz/save picture id=<name>}

This is the TikZ key that is used by \texttt{\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. \texttt{/tikz/check picture id}

There are circumstances where, behind the scenes, a \texttt{tikzpicture} is actually placed in a box and processed several times (often this involves \texttt{\mathchoice}). In such a situation, when defining nodes then the last one “wins” in that each node remembers the id of the last processed picture. However, only the one that is actually used has its location remembered on the page (since the others don’t have a position). This can lead to the situation whereby a node becomes disassociated from its picture and so using it for later reference fails. This key tries to get around that situation by checking the aux file to see if the current picture was actually typeset last time (by checking for the presence of the remembered location) and if it finds that it wasn’t, it quietly appends the string \texttt{discard-} to each node name. The idea being that the version of the picture that is actually typeset will not have this happen and so its nodes “survive”.

12. \texttt{/tikz/maybe define node=#1}

The previous key can lead to undefined nodes on the first time that the picture is processed. Using this key will ensure that the specified node is aliased to its \texttt{discard-} version providing it doesn’t already exist. This is purely to get rid of pointless error messages, and also should only be used in conjunction with \texttt{check picture id}.

Note that due to the order in which code gets executed, \texttt{check picture id} should be before any \texttt{maybe define node} keys.

13. \texttt{/tikz/if picture id=#1#2#3}

This is a key equivalent of the \texttt{\iftikzmark} command.

14. \texttt{/tikz/if tikzmark on current page=#1#2#3}

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

15. \texttt{/tikz/if tikzmark on page=#1#2#3#4}

This is a key equivalent of the \texttt{\iftikzmarkonpage} command.

16. \texttt{/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 \texttt{/tikz/next page vector}. The key \texttt{/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) \texttt{above}, \texttt{below}, \texttt{left}, \texttt{right}, and \texttt{ignore}. (The last one sets the vector to the zero vector.)

Previous versions of \texttt{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.

17. \tikzmark prefix=<prefix> and \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.

4.2 Pic and Scope Positioning

scope anchor, pic anchor, and surround pic.

These keys can be used to enable advanced positioning of scopes and pics. The standard positioning of a pic places its internal origin at the location specified on the \pic command. This is more limited than what is available to a node whereby any of the defined anchors can be placed at the given position. The key pic anchor allows a little more flexibility to pic positioning by allowing a pic anchor to be defined and used as the point to place at the given position.

When invoking the pic the key pic anchor={coordinate} can be used to specify a point inside the pic to use as the anchor. This point is evaluated inside the pic so if using a node then the node name should be specified as if inside the pic.

The node positioning syntax, things like below and below=5pt of, sets the anchor of the following node. Using pic anchor without a coordinate uses this anchor on the bounding box of the pic when positioning the pic.

Internally, this works by adjusting the location of the pic's surrounding scope. So the code can equally be used on scopes. For a scope, use the scope anchor version on the scope directly. The keys name and anchor can be used on the scope as if on a node with the same effect on the positioning.

The key surround pic saves the bounding box of the pic as if it were the boundary of a rectangular node, using the name of the pic as the name of the node.

This was inspired by the questions Anchoring TiKZ pics and Reposition Tikz Scope After Size Known.

4.3 Subnodes

\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 \texttt{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 \texttt{tikzpicture}) to access this information. Thus after \texttt{\node {a \subnode{a}{sub} node};} it is possible to use \texttt{a} as a node. The \texttt{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, \texttt{line width} affects the node size).

There are two important points to make about this. The first is that, as with all the \texttt{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 \texttt{fit} library. Using the above example, \texttt{\node[fit=(a),draw,inner sep=0pt] {};} would draw a rectangle around the word \texttt{sub} of exactly the same size as would appear had a normal node been created.

Using a sneaky trick with \texttt{mathchoice}, \texttt{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 got beyond that.

Note that because of the way that this works, the outer \texttt{tikzpicture} must have the \texttt{remember picture} option set.

### 4.4 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 \texttt{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.

- \texttt{\node\textit{save node}=\textit{name}}
  
  This is the key that indicates a node to be saved. The version with no argument is to be used directly in the keys for a node and it saves that node. With an argument then it saves a node that has been declared somewhere in the current tikzpicture (it may not always be convenient to issue the \texttt{save node} key directly on the node itself). Since the list is saved up to the end of the picture, this can be invoked before the node is defined.

- \texttt{\SaveNode\textit{group name}\{name\}}
This command is for outside a \texttt{tikzpicture} and saves the named node directly. The optional argument is a group name for saving to a group. If this is not specified then the node is saved to a file.

- \texttt{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.

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

- \texttt{save nodes to file}
  
  This is a \texttt{true/false} key which determines whether to save the node information to a file.

- \texttt{set saved nodes file name=<name>}
  
  This sets the file name for the saved nodes (the extension will be .\texttt{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.)

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

- \texttt{transform saved nodes}
  
  A particular use-case for restoring saved nodes is to safely include one \texttt{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

The \texttt{\textbackslash tikzmark} command has been used in numerous answers on \TeX-SX.

5.1 Basic Examples

A simple example of the \texttt{\textbackslash tikzmark} macro is the following.
\[ e^{i \frac{\pi}{2}} = i \]

This is an important equation.

\begin{itemize}
\item A first item,
\item A second item,
\item A third item.
\end{itemize}

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

\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:

\[
\tikzmarknode{a}{\sum_{k=1}^{n} k^{2}}^{\tikzmarknode{b}{(2)}}
\]

\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 \texttt{firstpicture.tex}:

\tikzset{tikzmark prefix=ex5-}
\begin{tikzpicture}[remember picture,overlay]
\draw[->] (txt) -- (a);
\draw[->] (a.south) to[out=-90,in=-45] (b.south east);
\end{tikzpicture}
\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:
6 Additional Libraries

Some of the more ambitious uses of \tikzmark involve a fair bit of extra code and so are worth gathering in to extra libraries of their own. These can be loaded via \usetikzmarklibrary.

At present, there are three libraries: one for code listings which works with the listings package, one for AMSMath equations, and one for highlighting.

6.1 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.

```latex
\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.
6.2 AMS Equation Environments

This is an experimental library. If the \texttt{amsmath} package has been loaded then issuing
\begin{tikzmarklibrary}[ams]
loads some code that places pseudo-nodes around the boxes that are used in
AMSMath's various equation alignment environments, such as \texttt{align} and \texttt{gather}.
These environments work by constructing boxes with each of the pieces of the
equations that are then put together into the grid. This library hooks in to the
unboxing code, before the box is typeset then it measures it and stores that information
in various macros as if it were a TikZ node. The aim is that this doesn’t disturb the placement,
but as far as TikZ is concerned then there is a node there that can be referred to later.

As it is experimental, even if this library is loaded then it isn’t automatically switched on. To do that, use either the \texttt{tikzmarkmath} environment or the \texttt{\tikzmarkmath} command. Each has an optional argument which is a pre-
fix for the node names (the default is \texttt{equation}). The node names are then
of the form \texttt{<prefix>-<number>}. The numbering is held in a counter called
\texttt{tikzmarkequation} and is reset when the command is invoked or the environ-
ment is started. As usual, redefining \texttt{\thetikzmarkequation} changes the styling
of the \texttt{<number>}.

To disable the marking, either end the environment or use \texttt{\endtikzmarkmath}.
The ending command explicitly removes the hook rather than rely on \texttt{TPX} group-
ings. It also prints out the number of nodes created to the log file and terminal.
This can be useful with figuring out which nodes to use, since the box that this li-
brary hooks into is used many times. For example, equation numbers are included
with this.

The box is also used when assembling a \texttt{\sqrt{3}}\texttt{\{4\}} command, and as that
uses \texttt{mathchoice} then there are more boxes created than used. So the count of number of nodes created can be more than are actually there.

\begin{tikzmarkmath}{pythagoras}
\begin{gather}
a^2 = b^2 + c^2
\end{gather}
\begin{gather}
a = \sqrt{b^2 + c^2}
\end{gather}
\end{tikzmarkmath}

\begin{tikzpicture}[remember picture, overlay]
\foreach \k in {1,2,3,7,8} {
  \draw[red] (pic cs:pythagoras-\k) -- ++(135:1)
  node[draw,red,circle,font=\tiny,above left] {\k};
  \node[draw,blue,fit=(pythagoras-\k),inner sep=0pt] {};
}
\end{tikzpicture}

\section{Highlighting}

I’ve returned to the highlighting library. The \LaTeX{} hook mechanism makes a couple of things possible that were tricky before.

The idea of the highlighting mechanism is to use two \texttt{\tikzmarks} to mark a start and end of a region to be highlighted. The region is considered to be formed by lines of text, with the first mark at the baseline of the start and the second at the baseline of the end.

The highlighting itself is done by inserting code in the shipout routine before the page itself is laid out. So the highlighting is on a separate layer to the text itself, which can be either behind or in front of the text layer. The hook mechanism also makes it relatively simple to support page breaks between the start and end of highlighting.

Since the highlighting is separate to the flow of the text, it doesn’t make sense to use an environment to mark the start and end of the highlighting so instead there are two commands: \texttt{\StartHighlighting[options]} and \texttt{\StopHighlighting}, or a single command \texttt{\Highlight[options]{text}} that just highlights the text. At the moment, nesting highlighting is not supported.

The optional argument to \texttt{\StartHighlighting} (or \texttt{\Highlight}) consists of key-value pairs that control the behaviour of the highlighted region. There are
particular keys in the /tikz/highlighter family which control the size of the highlighted region.

The keys are as follows:

- direction
- layer
- initial height
- initial depth
- initial offset
- final height
- final depth
- final offset
- left margin
- right margin
- height
- depth
- offset
- margin

The highlighting code draws a region which can be styled with standard TikZ keys, more of which in a moment. Although it is a single region, the intention is to simulate using an actual highlighter. The first key, direction, is used to draw the region as if the highlighter were used in a particular direction. The options are horizontal, vertical, or box. The default is horizontal.

The second key, layer, determines whether the highlighter is rendered on the background or foreground layer. Using the background layer puts the highlighting underneath the text, which will make the text easier to read. The foreground option puts the highlighting over the text, which can be used to fade the text. The default is background.

The shape of the region depends on a few things, such as whether the highlighting starts and ends on the same line.
The vertical regions and the box are defined similarly. With the vertical regions then the meaning of the height, depth, and offset are rotated 90°, and the vertical regions don’t stretch to the page boundaries. The box region is always a rectangle.

Once the region is defined, it can be styled using options directly on the \StartHighlighting or \Highlight command and by using the following styles:

- every highlight picture
- every <direction> highlight picture
- every <layer> highlight picture
- every highlight path
- every <direction> highlight path
- every <layer> highlight path
- highlight path
- <direction> highlight path
- <layer> highlight path
The picture keys are for the surrounding \texttt{tikzpicture}, while the path keys are for the path itself.

Lastly, a word about scoping the options. Since the code that actually renders the highlighting is processed when the page is shipped out, it may well be that the settings in force when the highlighting was defined have changed. The keys that adjust the size of the region (in the highlighter family) are saved at the moment of invocation but keys such as the colour or whether to fill or draw the path are not. Therefore, it is wise to use styles that persist to set the rendering styles.

The sun was shining on the sea, shining with all its might. \StartHighlighting[fill=cyan!50]
And this was very odd because it was the middle of the night. \StopHighlighting
The moon was up there sulkily because she thought the sun had no business to be there after the day was done. \StartHighlighting[fill=magenta!50] 
‘‘It’s very rude of him,’’ she said, ‘‘to come and spoil the fun.’’ \StopHighlighting
\noindent The sun was shining on the sea, shining with all its might. And this was very odd because it was the middle of the night. \StartHighlighting[fill=yellow!50] 
The moon was up there sulkily because she thought the sun had no business to be there after the day was done. ‘‘It’s very rude of him,’’ she said, ‘‘to come and spoil the fun.’’ \StopHighlighting.

7 Acknowledgements

The \texttt{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.

8 Implementation

8.1 Main Code

The \texttt{save nodes} code uses L\texttt{aTEX3}.

20
\ProvidesFile{tikzlibrarytikzmark.code.tex}[
2022/08/24
v1.15
TikZ library for marking positions in a document
\RequirePackage{expl3, l3keys2e, xparse}
\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\arabic{page}%
        }%
      }%
      \pgf@temp
    \fi%
  },
},

There are times when some code is executed and then discarded, such as in \mathchoice. This can seriously mess with how TikZ pictures are remembered as the last \pgfpictureid to be processed is the one that is used, but it is the one that is used that is recorded in the aux file. This isn’t particularly a tikzmark issue, but does come up from time to time with tikzmark as it’s all about remembering locations.

In actual fact, it only occurs with \tikzmarknode since the issue is about how nodes are associated with pictures.

The solution is to check to see if the \pgfpictureid has been recorded in the aux file and if it hasn’t, quietly prefix the node names with a discard term. This needs to be used after remember picture has been invoked. It probably messes with some other stuff so should only be used under controlled conditions, such as \tikzmarknode.

check picture id/.code={
  \ifpgfrememberpicturepositiononpage
    \@ifdefineline{\pdf@mark@pos\pgfpictureid}{
      \tikzset{%
        name prefix/.get=\tzmk@name@prefix, 
        name prefix/.prefix=discard-, 
        execute at end picture={%
          \tikzset{name prefix/.expand once=\tzmk@name@prefix}%
        },
      }%
    }%
  \fi
}
We also want a failsafe that quietly handles the case where the document hasn’t been compiled enough times (once) to get the information into the \texttt{aux} file. There will already be messages about needing reruns so we don’t need to add to that. We simply ensure that the node exists.

```latex
\begin{tikzpicture}
\node[style={
\ifpgfrememberpicturepositiononpage
\@ifundefined{pgf@sh@pi0\tikz@pp0\name{#1}}{\pgfnodealias{\tikz@pp0\name{#1}}{discard-\tikz@pp0\name{#1}}}{}
\fi
}};
\end{tikzpicture}
```

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

```latex
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)

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

Page handling

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

22
Prefix and suffix for tikzmark names, shamelessly borrowed from the main tikz code

tikzmark prefix/.initial=,
tikzmark suffix/.initial=,
tikzmark clear ixes/.style={
    tikzmark prefix={},
    tikzmark suffix={}
},

Tikzmarks can be used to adjust the position of a scope or pic so that an internally defined coordinate is used to locate the scope or pic.

The key used to adjust the location is scope anchor=(coordinate) for scopes and pic anchor=(coordinate) for pics, where coordinate is evaluated internally to the scope or pic, so can use node names.

scope anchor location/.initial={(0,0)},
scope anchor location/.default=@auto,
pic anchor/.style={
    scope anchor location=#1},
next pic/.append style={
}
The code that does the adjustment is added to the \texttt{pic} on its enclosing scope using the \texttt{every pic} key.

To install this code on a \texttt{pic}, we hook in to the \texttt{pic}'s enclosing scope using the \texttt{every pic} key. To avoid this bubbling down to \texttt{pics} within \texttt{pics}, we clear it once it has been executed. So any code that triggers this adjustment adds \texttt{adjust pic position} to the !next pic! style.

This code remembers the bounding box of a \texttt{pic}, saving it as if it were a node.


\tikzmarkppname

172 \def\tikzmarkppname#1{%
173 \csname pgfk/tikz/tikzmark prefix\endcsname#1%
174 \csname pgfk/tikz/tikzmark suffix\endcsname%
176 }%

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

177 \def\savepointas#1#2#3#4{%
178 \expandafter\gdef\csname save@pt@#1\endcsname{#2}%
179 \expandafter\gdef\csname save@pt@#1@offset\endcsname{%
180 \pgfpoint{#3}{#4}%
181 }%
182 \def\savepicturepage#1#2{%
183 \expandafter\gdef\csname save@pg@#1\endcsname{#2}%
184 }

\tikzmarkalias Alias a tikzmark to another name (used in tikzmarknode). The alias is saved to
the aux-file so that it is available prior to the definition. The private one doesn’t
use the prefix-suffix for greater internal flexibility. The public one does.

185 \def\@tikzmarkalias#1#2{%
186 \@ifundefined{save@pt@#2}{}{%
187 \pgf@node@gnamelet{save@pt@#1}{save@pt@#2}%
188 \pgf@node@gnamelet{save@pt@#1@offset}{save@pt@#2@offset}%
189 \protected@write\pgfutil@auxout{}{%
190 \string\savepointas%
191 {#1}{\csname save@pt@#2\endcsname}%
192 \expandafter\expandafter\expandafter
193 \@gobble\csname save@pt@#2@offset\endcsname%
194 }%
195 }%
196 }
197 \def\tikzmarkalias#1#2{%
198 \@tikzmarkalias{\tikzmark@pp@name{#1}}{\tikzmark@pp@name{#2}}%
199 }

\tmk@labeldef Auxiliary command for the coordinate system.

200 \def\tmk@labeldef#1,#2\@nil{%
201 \edef\tmk@label{\tikzmark@pp@name(#1)}{%
202 \edef\tmk@def{#2}%
203 }
204 \edef\tmk@label{\tikzmark@pp@name(#1)}{%
205 \tmk@def{#2}%
206 }

\pic This defines the new coordinate system.

207 \tikzdeclarecoordinatesystem{pic}{%
208 \pgfutil@in@,{#1}%
209 \ifpgfutil@in@
210 \tmk@labeldef{#1,2\@nil}{%
211 \edef\tmk@labeldef{\tikzmark@pp@name(#1)}{%
212 \edef\tmk@def{#2}%
213 }
214 \else
215 \tmk@labeldef{#1,(0pt,0pt)}\@nil
216 \fi
217 }%
The active/non-active semi-colon is proving somewhat hazardous to \texttt{tikzmark} (see `[Tikzmark and french seem to conflict](http://example.com)` 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][{}]{\tikzset{external/export next/.try=false}\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@resetexpandcount%  
\tikz@scan@one@point\pgfutil@firstofone#2\relax
\pgf@pos@transform{\pgf@x}{\pgf@y}%  
\protected@write\pgfutil@auxout{%  
\string\savepointas%  
\{\tikzmark@pp@name{#1}\}{\pgfpictureid}{\the\pgfx@x}{\the\pgf@y}}%  
\end{verbatim}

And finally, the ultimate invoker:

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

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

\begin{verbatim}
\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}{Opt}{Opt}}%  
%  \egroup  
%
%  \ifclassloaded{beamer}{%  
%  \renewcommand{%  
%  \{\tikzmark@outside}{}%  
%  \only#3{\beameroriginal{\tikzmark@outside}{#1}{#2}}%  
%  \}  
%  \renewcommand{%  
%  \{\tikzmark@inside}{}%  
%  \only#3{\beameroriginal{\tikzmark@inside}{#1}{#2}}%  
%  \end{verbatim}
If beamer is loaded, add a suffix based on the frame number

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

\iftikzmarkonpage

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

The `\pgfmark` applies the current tikzmark prefix/suffix. The current node prefix/suffix is applied by using the `name=` key.
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.

\def\tikzmarknode[#1][#2]{%
This macro takes a name and a box. It pretends that there is a tight-fitting rectangular PGF node around that box with the given name, and saves the required information so that that node can be used later on in a tikzpicture drawing.
It does not actually build a node, and it doesn’t create a TikZ drawing. Rather, it measures the box and uses that information to define the various macros that store the information about the node.

Apart from assigning a load of macros, it does also place a \texttt{\pgfmark} just before the box. This is needed to be able to locate the node on the page.

The command is defined with an @ because it is more likely to be used in other packages than by a user.

```latex
543 \def\tikzmark@box#1#2{% 
544 \begingroup 
545 \pgfmark{#1}% 
546 \let\pgfnodeparttextbox=#2% 
547 \edef\pgfpictureid{pgfid\the\pgf@picture@serial@count}% 
548 \def\tikz@fig@name{#1}% 
549 \pgfpointorigin 
550 \advance\pgf@x by .5\wd\pgfnodeparttextbox 
551 \advance\pgf@y by .5\ht\pgfnodeparttextbox 
552 \advance\pgf@y by -.5\dp\pgfnodeparttextbox 
553 \pgftransformshift{}% 
554 \setbox\@tempboxa=\hbox\bgroup 
555 {\tikzset{ 
\inner sep=0pt, 
minimum size=0pt, 
outer sep=0pt, 
anchor=base
}}% 
556 \let\pgf@sh@savedmacros=\pgfutil@empty% MW 
557 \let\pgf@sh@savedpoints=\pgfutil@empty 
558 \def\pgf@sh@shape@name{rectangle}% CJ % TT added prefix!
559 \pgf@sh@anchor{rectangle}{center}%
560 \pgf@x=-\pgf@x
561 \pgf@y=-\pgf@y
562 }\expandafter\pgfsavepgf@process
563 \csname pgf@sh@anchor{rectangle}{center}\endcsname{% FIXME : this is double work!
564 \pgf@sh@anchor{rectangle}{center}%
565 \expandafter\pgf@sh@@temp\expandafter{\pgf@sh@savedpoints}
566 \edef\pgf@sh@@temp{\endcsname{%
\pgf@sh@savedpoints}
```

33
The save node code is written in \LaTeX.3.
This is how we handle return values from functions.

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.

We’ll need a couple of spare token lists.

Another useful token list.

This token list is used for our current node group name.

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

Has to be global as we’re often in a group.

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

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

Key interface for setting some of the options.

Dimensions and token lists for shifting.

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

Set up a stream for saving the nodes data to a file.
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
\cs_new_nopar:Npn \tikz_scan_point:n #1
{\tikz@scan@one@point\pgfutil@firstofone#1\relax}
\cs_new_nopar:Npn \tikz_scan_point:NNn #1#2#3
{\tikz@scan@one@point\pgfutil@firstofone#3\relax\dim_set_eq:NN #1 \pgf@x\dim_set_eq:NN #2 \pgf@y}
\makeatother
\cs_generate_variant:Nn \tikz_scan_point:n {V}
\cs_generate_variant:Nn \tikz_scan_point:NNn {NNV}

\process_node: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 the node name to be saved.
\cs_new_nopar:Npn \__sn_process_node:n #1
{\group_begin:Clear our token list\tl_clear:N \l__sn_tmpa_tlSet the centre of the picture\tikz_scan_point:Nn \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 {
\dim_use:N \l__sn_x_dim \dim_use:N \l__sn_y_dim}
\tl_if_exist:cT {pgf@sh@ms@#1}
{
Test to see if the node has been defined
\tl_set:cn {pgf@sh@XX@nodename}
\clist_map_inline:nn {ns,np,ma,pl}
{\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.
Our token list will look like:
\tl_set:cn {pgf@sh@XX@nodename} <current contents of that macro>
This part puts the \tl_set:cn {pgf@sh@XX@nodename} in place
\tl_put_right:Nn \l__sn_tmpa_tl {\tl_gset:cn {pgf@sh@##1@ \tikz_set_node_name:n{#1} }}
\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_centre_tl {pgf@sh@nt@#1}
\tl_put_right:Nx \l__sn_tmpa_tl {
\tl_set:cn {pgf@sh@nt@0 \tikz_set_node_name:n{#1}}
\compose_transformations:NVv \l__sn_centre_tl {pgf@sh@nt@0#1}
\tl_put_right:Nx \l__sn_tmpa_tl {{\exp_not:v \l__sn_tmpb_tl}}
\tl_put_right:Nx \l__sn_tmpa_tl {
\transform_node:Nn \l__sn_transformation_tl {
\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_set_eq:NN \g__sn_output_tl \l__sn_tmpa_tl
\group_end:
}
\cs_new_protected_nopar:Npn \process_node:Nn #1#2
{ \__sn_process_node:n {#2} \tl_set_eq:NN #1 \g__sn_output_tl \tl_gclear:N \g__sn_output_tl }
\cs_new_protected_nopar:Npn \process_gnode:Nn #1#2
{ \__sn_process_node:n {#2} \tl_gset_eq:NN #1 \g__sn_output_tl \tl_gclear:N \g__sn_output_tl }

\save_nodes_to_list:nn
Save the nodes to a list, given a key
\cs_new_nopar:Npn \save_nodes_to_list:nn #1#2
{ \tl_clear:N \l__sn_tmpa_tl \clist_map_inline:nn {#2} { \process_node:Nn \l__sn_tmpb_tl {##1} \tl_put_right:NV \l__sn_tmpa_tl \l__sn_tmpb_tl } \prop_gput:NnV \g__sn_prop {#1} \l__sn_tmpa_tl }

\save_nodes_to_file:n
Save the nodes to a file
\cs_generate_variant:Nn \iow_now:Nn {NV} \cs_new_nopar:Npn \save_nodes_to_file:n #1
{ \sn_open_stream: \clist_map_inline:nn {#1} { \process_node:Nn \l__sn_tmpa_tl {##1} \iow_now:Nx \g__sn_stream \iow_newline: \exp_not:V \l__sn_tmpa_tl } } \cs_generate_variant:Nn \save_nodes_to_list:nn {VV, Vn} \cs_generate_variant:Nn \save_nodes_to_file:n {V}

\restore_nodes_from_list:n
Save the nodes to a file
\cs_new_nopar:Npn \restore_nodes_from_list:n #1
{ 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 \l__sn_tmpa_tl }
I think the PGF Manual might be incorrect. It implies that the matrix is
stored row-major, but experimentation implies column-major.
That is, \( \begin{pmatrix} a & c \\ b & d \end{pmatrix} \) is:

\[ \begin{pmatrix} a & c \\ b & d \end{pmatrix} \]
\transform_node:Nn
\cs_new_nopar:Npn \transform_node:Nn #1#2
{\compose_transformations:cVv {pgf@sh@nt@#2} #1 {pgf@sh@nt@#2}}
\set_transform_from_node:n
\cs_new_nopar:Npn \set_transform_from_node:n #1
{\tl_set_eq:Nc \l__sn_transformation_tl {pgf@sh@nt@#1}
\tikz_scan_point:NNn \l__sn_x_dim \l__sn_y_dim {{#1.center}}
\dim_set:Nn \l__sn_x_dim \l__sn_x_dim - \tl_item:cn {pgf@sh@nt@#1}{5}
\dim_set:Nn \l__sn_y_dim \l__sn_y_dim - \tl_item:cn {pgf@sh@nt@#1}{6}
\compose_transformations:NnV \l__sn_transformation_tl {\begin{array}{cc}{1}{0}{0}{1}{\dim_use:N \l__sn_x_dim}{\dim_use:N \l__sn_y_dim}\end{array}} \l__sn_transformation_tl

\cs_generate_variant:Nn \set_transform_from_node:n {v}

Set the \texttt{TikZ} keys for access to the above commands.

\tikzset{
  set~ saved~ nodes~ file~ name/.code={\tl_gset:Nx \g__sn_filename_tl {#1}},
  transform~ saved~ nodes/.code={\set_transform_from_node:v \tikz@last@fig@name},
  set~ node~ group/.code={\tl_set:Nn \l__sn_group_tl {#1}\pgfkeysalso{\execute~ at~ end~ scope={\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\tl_if_eq:nnTF {#1} {false}{\bool_set_true:N \l__sn_file_bool\pgfkeysalso{\execute~ at~ end~ scope={\maybe_save_nodes:\}}\pgfkeysalso{\execute~ at~ end~ scope={\maybe_save_nodes:\}}}}},

\save~ node/.code={\tl_if_eq:nnTF {#1} \pgfkeysnovalue{\tikz_fig_must_be_named:\pgfkeysalso{\append~ after~ command={\pgfextra{\clist_gput_right:Nv \g__sn_nodes_clist \tikz@last@fig@name}}}}}}

Append current node or named node to the list of nodes to be saved
{\clist_gput_right:Nn \g__sn_nodes_clist {#1}}

\Restore nodes from file
\Restore nodes from file/.code={
  \bool_if:NTF \g__sn_preamble_bool {
    \restore_nodes_from_file:x {#1}
  }
  {
    \tikz_fig_must_be_named:
    \pgfkeysalso{append after command={
      \pgfextra{
        \scope
        \split_argument:Nv \tikzset \restore_nodes_from_file:x {#1}
        \endscope
      }
    }}
  }
},
\Restore nodes from list
\Restore nodes from list/.code={
  \tikz_fig_must_be_named:
  \pgfkeysalso{append after command={
    \pgfextra{
      \scope
      \split_argument:Nv \tikzset \restore_nodes_from_list:n {#1}
      \endscope
    }
  }}
},
\cs_generate_variant:Nn \clist_gput_right:Nn {Nv}
\split_argument:Nv
\split_argument:Nv \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:NN {#1}{#2}{#3}
}\split_argument_aux:NNp
\split_argument_aux:NNp \tl_set:Nx \l__sn_tmpa_tl {\tl_head:n {#3}}
8.2 Listings

From http://tex.stackexchange.com/q/79762/86
1028 \@ifpackage{listedings}{%

\iflst@linemark
A conditional to help with placing the mark at the first non-whitespace character.
Should be set to true so that we notice the first line of the code.
1029 \nevi\iflst@linemark
1030 \lst@linemarktrue

EveryLine
This hook places the mark at the start of the line.
1031 \lst@AddToHook{EveryLine}{%
1032 \begingroup
1033 \advance\c@lstnumber by 1\relax
1034 \pgfmark{line-\lst@name-\the\c@lstnumber-start}\
1035 \endgroup
1036 }

EOL
This hook places the mark at the end of the line and resets the conditional for
placing the first mark.
1037 \lst@AddToHook{EOL}{\pgfmark{line-\lst@name-\the\c@lstnumber-end}\
1038 \global\lst@linemarktrue
1039 }

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.
1040 \lst@AddToHook{OutputBox}{%
1041 \iflst@linemark
1042 \pgfmark{line-\lst@name-\the\c@lstnumber-first}\
1043 \global\lst@linemarkfalse
1044 \fi
1045 }

\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.
1046 \def\tikzmk@lst@fnum#1\relax#2\@STOP{%
1047 \def\@test{#2}\
1048 \ifx\@test\@empty
1049 \def\tikzmk@lst@start{0}\
1050 \else
1051 \@tempcnta=#1\relax
1052 \advance\@tempcnta by -1\relax
1053 \def\tikzmk@lst@start{\the\@tempcnta}\
1054 \fi
1055 }

Init
Adds a mark at the start of the listings environment.
1056 \lst@AddToHook{Init}{%
1057 \expandafter\tikzmk@lst@fnum\lst@firstnumber\relax\@STOP
1058 \pgfmark{line-\lst@name-\tikzmk@lst@start-start}\
1059 }
1060 }}%
\PackageError{tikzmark listings}{The listings package has not been loaded.}{
1061 {The listings package has not been loaded.}{
1062 }
1063 }
This tikzmark library defines a routine that puts a pseudo-node (using \tikzmark@box) around all the pieces used in constructing the various math environments that the AMS Math package provides, such as gather and align. All of these (and their labels) work by putting various pieces into a box and then typesetting that box in the cells of an halign. By using \tikzmark@box, this can be infiltrated to put nodes around each of those boxes as it is placed.

\@ifpackageloaded{amsmath}{%

\subsection*{AMS Math}

\tikzmarkmath\ Defines an environment in which any AMS mathematical aligned environments get nodes around each piece of their contents.

Start by saving the original \boxz@ command.

\let\tikzmark@ams@boxz@=\boxz@

We'll need a counter to keep track of the nodes.

\newcounter{tikzmarkequation}

The nodes will be labelled <name>-<number>. By default the name is equation but this can be customised.

\def\tikzmark@ams@name{equation}

This is the substitute command. I don't know if the \ifmeasuring@ actually does anything, but it's here just in case at the moment.

\def\tikzmark@boxz@{%
\ifmeasuring@
\tikzmark@ams@boxz@
\else
\stepcounter{tikzmarkequation}%
\tikzmark@box{\tikzmark@ams@name-\the\value{tikzmarkequation}}{\z@}%
\fi
}

This is the environment that sets the node name and swaps out the box code. At the end of the environment we swap back the code so that the commands can be used as standalone \tikzmarkmath and \endtikzmarkmath in occasions when it isn't appropriate to use an environment (for example, if it crosses sections, or if it is wanted to turn on this feature for an entire document). At the end of the environment, the number of nodes is written out to the terminal and log file to make it easier to keep track.

\newenvironment{tikzmarkmath}{\def\tikzmark@ams@name{#1} \setcounter{tikzmarkequation}{0} \let\boxz@=\tikzmark@ams@boxz@\%}{\let\boxz@=\tikzmark@ams@boxz@ \message{Tikzmark math environment \tikzmark@ams@name\space had \the\value{tikzmarkequation} nodes in it} \%}
8.4 Highlighting

An early use of \tikzmark was to add highlighting to text by drawing over or under the text between two \tikzmarks, for example the question \texttt{How to "highlight" text/formulas with \tikz?}.

I was never totally happy with the overall mechanism, so didn’t include it in the main \tikzmark package. Recently, I had occasion to revisit it and by using the new \LaTeX3 hook facility I got something that I was sufficiently happy with to add to the main package.

The key idea is to hook into the \texttt{shipout/background} routine to insert the highlighting behind the text. This allows us to draw the highlighting before the page is laid out and so is under the text.

\LaTeX3 makes life just that little bit easier.

\ExplSyntaxOn

Since the code that draws the highlighting will probably be very separate from the code that defines it, when storing the highlighting code then we want to expand the \tikzmark full name.

\cs_new_protected_nopar:Npn \tikzmark_fix_name:Nn #1#2
{\tl_set:Nx #1 \tikzmark@pp@name{#2}}

\StartHighlighting, \EndHighlighting, \Highlight

These are the user interfaces for highlighting a section. The first command inserts the drawing code into the relevant hook and places a \tikzmark at the current location. The second command indicates when the highlighting should stop. The third is a short cut for highlighting its argument.

These are commands rather than an environment to allow it to span, for example, different parts of an aligned equation.

\cs_new_protected_nopar:Npn \tikzmark_bake_highlighter:N #1
{\tl_clear:N #1 \clist_map_inline:nn {direction,layer}{/tikz/highlighter/##1=\pgfkeysvalueof{/tikz/highlighter/##1},}
\clist_map_inline:nn { initial~ height, initial~ depth, initial~ offset, final~ height, final~ depth, final~ offset, left~ margin, right~ margin, top~ margin, bottom~ margin, }
{
\tl_put_right:Nx #1 { /tikz/highlighter/#1=\dim_eval:n {\pgfkeysvalueof{/tikz/highlighter/#1}}, }
}
\cs_new_protected_nopar:Npn \tikzmark_start_highlighting:n #1
{
\int_gincr:N \g__tikzmark_highlighter_int
\tl_set:Nx \l__tikzmark_highlighter_name_tl {\tl_use:N \g__tikzmark_highlighter_tl \int_use:N \g__tikzmark_highlighter_int}
\tl_set:Nn \l__tikzmark_tmpb_tl {every~ highlighter/.try,}
\tikzmark{highlight-start-\tl_use:N \l__tikzmark_highlighter_name_tl}
\cs_new_protected_nopar:Npn \tikzmark_end_highlighting:
{
\tl_set:Nx \l__tikzmark_highlighter_name_tl {\tl_use:N \g__tikzmark_highlighter_tl \int_use:N \g__tikzmark_highlighter_int}
\tikzmark{highlight-end-\tl_use:N \l__tikzmark_highlighter_name_tl}
\NewDocumentCommand \StartHighlighting {O{}}
{\tikzmark_start_highlighting:n {#1}}
}}
The following code inserts the drawing command into the shipout hook.

We need an ordinary colon, rather than a \LaTeX3 one

\tl_const:Nx \_\_\_tikzmark_colon_tl
\char_generate:nn \{:'\} \{12\}
\cs_generate_variant:Nn \hook_gput_next_code:nn {nV}
\cs_new_protected_nopar:Npn \tikzmark_highlight_or_shunt:nnnn #1#2#3#4

First, test to check if the tikzmarks are actually defined yet, if not then bail out.
\bool_lazy_all:nT
\{\tl_if_exist_p:c \{save@pt@\tikzmark@pp@name{#2}}\}
\tl_if_exist_p:c \{save@pg@\tl_use:c\{save@pt@\tikzmark@pp@name{#2}}\}
\tl_if_exist_p:c \{save@pg@\tl_use:c\{save@pt@\tikzmark@pp@name{#3}}\}
\tl_if_exist_p:c \{save@pg@\tl_use:c\{save@pt@\tikzmark@pp@name{#3}}\}
\} \}

Okay, so all the tikzmarks are defined. Now see if we’re on the right page. Is our
start tikzmark in the future?
\int_compare:nTF
\{\tl_use:c \{save@pg@\tl_use:c\{save@pt@\tikzmark@pp@name{#2}}\}
\the\value{page}
\} \}

It is, so we just punt our highlighting down the line
\hook_gput_next_code:nn \{#1\} \{
\tikzmark_highlight_or_shunt:nnnn \{#1\}{#2}{#3}{#4}
\}
\}
\}

It isn’t, so we have some highlighting to do. We need to build our highlighting
code.
\tl_set:Nn \l__tikzmark_tmpa_tl \{#4\}
Is our starting tikzmark on this page?
\int_compare:nTF
{\tl_use:c {save@pg@\tl_use:c{save@pt@\tikzmark@pp@name(#2)}} = \the\value{page} }
{ }

It is, so we use the starting tikzmark as our first coordinate.

\tl_put_right:Nx \l__tikzmark_tmpa_tl
{ }

\tl_use:N \c__tikzmark_colon_tl #2 }
}
}

If isn’t, so we use the north west corner of the page

\tl_put_right:Nn \l__tikzmark_tmpa_tl
{ }

\tl_use:N \c__tikzmark_colon_tl

\tl_use:N \c__tikzmark_colon_tl

\tl_use:N \c__tikzmark_colon_tl

\tl_put_right:Nn \l__tikzmark_tmpa_tl
{ }

\tl_use:N \c__tikzmark_colon_tl #3 }
}
}

Is our ending tikzmark on this page?
\int_compare:nTF
{\tl_use:c {save@pg@\tl_use:c{save@pt@\tikzmark@pp@name(#3)}} = \the\value{page} }
{ }

It is, so we use the ending tikzmark as our second coordinate.

\tl_put_right:Nx \l__tikzmark_tmpa_tl
{ }

\tl_use:N \c__tikzmark_colon_tl #3 }
}
}

It isn’t, so we use the south east corner of the page, and we have to shunt the code to the next page.

\tl_put_right:Nn \l__tikzmark_tmpa_tl
{ }

{ }


We've built our highlighting code, now's time to execute it.

```latex
\hook_gput_next_code:nn {#1} {
  \tikzmark_highlight_or_shunt:nnnn {#1}{#2}{#3}{#4}
}
```

\tikzmark_process_highlighting:nn #1#2
{
\pgfkeys{/tikz/highlighter/configuration/.activate~ family}
\pgfkeysfiltered{/tikz/.cd,highlighter/direction,highlighter/layer,#1}
\tikzmark_fix_name:Nn \l__tikzmark_start_tl {highlight-start-#2}
\tikzmark_fix_name:Nn \l__tikzmark_end_tl {highlight-end-#2}
\tl_set:Nx \l__tikzmark_tmpa_tl \pgfkeysvalueof{/tikz/highlighter/direction}
\tl_clear:N \l__tikzmark_tmpb_tl
\tl_clear:N \l__tikzmark_tmpc_tl
\tl_if_eq:NnTF \l__tikzmark_tmpa_tl {vertical}
{
  \tl_put_right:Nn \l__tikzmark_tmpb_tl
  \vl@draw
}
\tl_if_eqNnTF \l__tikzmark_tmpa_tl {box}
{
  \tl_put_right:Nn \l__tikzmark_tmpb_tl
  \box@draw
}
\tl_if_eqNnTF \l__tikzmark_tmpa_tl {horizontal}
{
  \tl_put_right:Nn \l__tikzmark_tmpb_tl
  \hl@draw
}
\tl_set:Nx \l__tikzmark_tmpa_tl \pgfkeysvalueof{/tikz/highlighter/layer}
\tl_set:Nn \l__tikzmark_tmpc_tl
```

50
The command that draws the horizontal highlighter or fader. This fills a shape
determined by two coordinates assumed to be (in effect) on the baseline of the
start and end of the region to be highlighted.

\def\hl@draw#1#2#3{\pgfkeys{/tikz/highlighter/configuration/.activate family}\
\pgfkeysfiltered{/tikz/.cd,highlighter/direction,highlighter/layer,#1}\
\begin{tikzpicture}[
\remember picture,\
\overlay,\
\highlight picture action,\
#1,\
\]}
\cs_generate_variant:Nn \tikzmark_process_highlighting:nn {nV,VV}
\ExplSyntaxOff
\begin{tikzpicture}
  \node \texttt{\textbackslash path[} \texttt{\textbackslash highlight action,}\n  \texttt{\#1 \textbackslash]} (\texttt{start }-| \texttt{west}) \texttt{rectangle} (\texttt{end});
  \else
  \texttt{\textbackslash path (#2)} \texttt{++(-1*pgfkeysvalueof{/tikz/highlighter/initial offset},}\n  \texttt{pgfkeysvalueof{/tikz/highlighter/initial height}) \texttt{coordinate (tl); \textbackslash path (#2)} \texttt{++(-1*pgfkeysvalueof{/tikz/highlighter/initial offset},}\n  \texttt{-1*pgfkeysvalueof{/tikz/highlighter/initial depth}) \texttt{coordinate (start); \textbackslash path (#3)} \texttt{+pgfkeysvalueof{/tikz/highlighter/final offset},}\n  \texttt{-1*pgfkeysvalueof{/tikz/highlighter/final depth}) \texttt{coordinate (end); \path (#3)} \texttt{+pgfkeysvalueof{/tikz/highlighter/final offset},}\n  \texttt{pgfkeysvalueof{/tikz/highlighter/final height}) \texttt{coordinate (mr); \textbackslash path[} \texttt{\textbackslash highlight action,}\n  \texttt{\#1 \textbackslash]} (\texttt{start}) \texttt{-- (tl) \texttt{-- (tl -| east) \texttt{-- (mr -| east) \texttt{-- (mr) -- \texttt{(end)} \texttt{-- (end -| west) \texttt{-- (start -| west) \texttt{-- cycle; \textbackslash fi \textbackslash fi \end{tikzpicture}}\texttt{\textbackslash def\texttt{box2draw#1#2#3}}}\texttt{\textbackslash pgfkeys{/tikz/highlighter/configuration/.active family}}\texttt{\textbackslash pgfkeysfiltered{/tikz/.cd,highlighter/direction,highlighter/layer,#1}}\texttt{\begin{tikzpicture} remember picture, overlay, highlight picture action, \texttt{\#1, \textbackslash} \texttt{\end{tikzpicture}}\texttt{\relax \pgf@xa=\pgf@x \texttt{\relax \pgf@xb=\pgf@x \texttt{\relax \relax 53}}}
In this one the region is defined vertically.

\def\vl@draw#1#2#3{\pgfkeys{/tikz/highlighter/configuration/.activate family}\pgfkeysfiltered{/tikz/.cd,highlighter/direction,highlighter/layer,#1}\begin{tikzpicture}[\remember picture,overlay,highlight picture action,#1]\tikz@scan@one@point\pgfutil@firstofone(#2)\relax\pgf@ya=\pgf@y\pgf@xa=\pgf@x\tikz@scan@one@point\pgfutil@firstofone(#3)\relax\pgf@yb=\pgf@y\pgf@xb=\pgf@x\pgfmathsetlength\pgf@y{% \pgfkeysvalueof{/tikz/highlighter/initial offset}\pgfmathsetlength\pgf@y{% \pgfkeysvalueof{/tikz/highlighter/initial height}}}\pgfmathsetlength\pgf@y{% \pgfkeysvalueof{/tikz/highlighter/final offset},-1*\pgfkeysvalueof{/tikz/highlighter/final depth}}coordinate (start);\path (#3)++(\pgfkeysvalueof{/tikz/highlighter/initial height},\pgfkeysvalueof{/tikz/highlighter/initial offset})coordinate (end);\path[\hlaction] (start) rectangle (end);\end{tikzpicture}
\pgfkeysvalueof{/tikz/highlighter/initial offset})
coordinate (start);
\path (#3) ++(-1*\pgfkeysvalueof{/tikz/highlighter/final depth},
-1*\pgfkeysvalueof{/tikz/highlighter/final offset})
coordinate (end);
% else
\path (#2) ++(-1*\pgfkeysvalueof{/tikz/highlighter/initial depth},
\pgfkeysvalueof{/tikz/highlighter/initial offset})
coordinate (start);
% else
\path (#2) ++(\pgfkeysvalueof{/tikz/highlighter/final height},
-1*\pgfkeysvalueof{/tikz/highlighter/final offset})
coordinate (end);
\fi
% \path[
  highlight action,
  #1]
(start) rectangle (end);
% else
\path (#2) ++(\pgfkeysvalueof{/tikz/highlighter/initial height},0)
coordinate (tr);
\path (#2) ++(0,\pgfkeysvalueof{/tikz/highlighter/initial offset})
coordinate (start);
% else
\path (#2) ++(-1*\pgfkeysvalueof{/tikz/highlighter/initial depth},0)
coordinate (tl);
\path (#3) ++(\pgfkeysvalueof{/tikz/highlighter/final height},0)
coordinate (br);
\path (#3) ++(0,-1*\pgfkeysvalueof{/tikz/highlighter/final offset})
coordinate (end);
% else
\path (#3) ++(-1*\pgfkeysvalueof{/tikz/highlighter/initial depth},0)
coordinate (bl);
\path (#3)
++(0,-1*\pgfkeysvalueof{/tikz/highlighter/final offset})
coordinate (end);
% else
\path (#3)
++(0,-1*\pgfkeysvalueof{/tikz/highlighter/final offset})
coordinate (end);
% else
\path (#3)
++(-1*\pgfkeysvalueof{/tikz/highlighter/initial depth},0)
coordinate (bl);
% \tikz@scan@one@point\pgfutil@firstofone(#2)\relax
These set various options.
\tikzset{%
/tikz/highlighter/.is family,
/tikz/highlighter/.unknown/.code={%
\let\tk@searchname=\pgfkeyscurrentname%
\pgfkeysalso{%
/tikz/\tk@searchname=#1
}
/tikz/\tk@searchname=#1
}
},
every highlight path/.style={
  fill=yellow!50,
  rounded corners,
},
every foreground highlight path/.style={
  fill opacity=.5,
},
highlight picture action/.style={
  every highlight picture/.try,
  every \pgfkeysvalueof{/tikz/highlighter/direction} highlight picture/.try,
  every \pgfkeysvalueof{/tikz/highlighter/layer} highlight picture/.try,
},
highlight action/.style={
  every highlight path/.try,
  every \pgfkeysvalueof{/tikz/highlighter/direction} highlight path/.try,
  every \pgfkeysvalueof{/tikz/highlighter/layer} highlight path/.try,
  highlight path/.try,
  \pgfkeysvalueof{/tikz/highlighter/direction} highlight path/.try,
  \pgfkeysvalueof{/tikz/highlighter/layer} highlight path/.try,
},
/tikz/highlighter/.cd,
layer/.initial=background,
direction/.default=horizontal,
layer/.default=background,
initial height/.initial=\baselineskip,
initial depth/.initial=.5\baselineskip,
initial offset/.initial=.5\baselineskip,
final height/.initial=\baselineskip,
final depth/.initial=.5\baselineskip,
final offset/.initial=.5\baselineskip,
left margin/.initial=.5\baselineskip,
right margin/.initial=.5\baselineskip,
top margin/.initial=.5\baselineskip,
bottom margin/.initial=-.5\baselineskip,
height/.style={
  initial height=#1,
  final height=#1
},
deepth/.style={
  initial depth=#1,
  final depth=#1
},
offset/.style={
  initial offset=#1,
  final offset=#1
},
margin/.style={
  left margin=#1,
  right margin=#1,
  top margin=#1,
  bottom margin=#1,
},
/tikz/highlighter/configuration/.is family,
/tikz/highlighter/direction/.belongs to family=/tikz/highlighter/configuration,
/tikz/highlighter/layer/.belongs to family=/tikz/highlighter/configuration,
\def\page@node{
  \path (current page.north west)
  ++(\offset + 1in + \oddsidemargin + \leftskip,
  -\offset - 1in - \topmargin - \headheight - \headsep)
  node[
  minimum width=\textheight - \leftskip - \rightskip,
  minimum height=\textheight,
  anchor=north west,
  line width=0mm,
  inner sep=0pt,
  outer sep=0pt,
  ] (page) {};
}