The Gamebooklib Package

Robert J Lee
latex@rjlee.homelinux.org
August 24, 2021

Abstract
This package provides macros and environments to allow the user to typeset a series of cross-referenced, numbered “entries”, shuffled into random order.

1 Introduction

This package was written to allow the typesetting of gamebooks.

A gamebook is a book divided into “entries” (which may be a single paragraph or longer), each with a sequentially numbered value. At the end of each entry, a link to one or more other numbered entries is given; the reader selects one and they follow through the story in this order. gamebooks traditionally start at the entry numbered “1” and some gamebooks have multiple endings, with the final section representing a success or victory for the reader.

In particular, this package handles two technical challenges which are tricky to solve with \LaTeX directly: writing the “entries” in order but presenting them in a randomised order; and presenting footnotes at the end of each “entry”, numbered in the order in which they appear, not the order in which they are written.

Aside from shuffling the paragraphs and fixing the handling of footnotes, this package provides little help for the actual typesetting of gamebook entries. For this, please consider using André Miede’s gamebook package, available on CTAN\(^1\). That package is orthogonal to this one; both may be used together.

Terminology

For the purposes of this document, the term entry means a numbered section of text. This term helps describe the common format of a gamebook, while avoiding confusion with the terms paragraph and a section, both of which already have a clear definition.

The term gamebook means a book consisting of numbered sections, typically presented in a non-linear order with a tree or graph of possible reading options.

\(^*\)This document corresponds to Gamebooklib Gamebook, dated 2021/08/24.
\(^1\)https://www.ctan.org/pkg/gamebook
Usage

To load the class, use \usepackage[options]{gamebook}

The class options are described below; \footnotes is normally a good choice, although there are some limitations.

The user simply writes entries like this inside the document:

\begin{gentry}{codea}
This is the first entry
\end{gentry}

Turn to \turnto{codeb}
\begin{gentry}{codeb}
This is the second and final entry.\par
To play again, turn to \turnto{codea}
\end{gentry}

% Output:
\hegentries

Here, two entries are defined, and then output.
The first entry is given a label of gentry:codea, and the \turnto{codea} in the second entry generates a link back to that label.

As entry is a common term likely to conflict with other environments, the environment to declare an entry has been named gentry (gamebook entry) instead.

Sometimes it is convenient, when writing an entry, to give it a fixed index in the text. The first optional argument in the second entry — [123] — fixes the entry at that position during shuffling. This is implemented by iterating over each entry and swapping it with that index. Duplicates are not currently checked for, but they would not be lost; if two entries are defined with the same fixed index, one will end up somewhere else in the text.

The first and last entries are automatically fixed without needing to specify the optional number.

Specifying the optional number as a value greater than or equal to the number of entries, or less than 2, is not recommended. (Yes, this example used 123, which is beyond the number of entries. This was done to show the syntax).

The second optional argument is a title to be displayed alongside the numerical value.

Finally, the command \thegentries will output all entries defined so far, handling shuffling and footnotes.

\hentryheader{(counterIdx)}{(fixedIdx)}{(code)}{(title)}

Users may also want to redefine \gentryheader to change how the entries are displayed. The initial version is quite basic and intended for debugging; users should use \renewcommand to change it to something more pleasant.

Here's one suggestion, using packages from CTAN. It works well if you have either a title, or at least 2 lines of text before the first paragraph break inside the
gentry.
   It also requires near the start of the document:
   \usepackage{lettrine}
   \usepackage{etoolbox}
   \usepackage{color}

And this anywhere before \thegentries:
   \renewcommand\LettrineFontHook{\color[gray]{0.5}\fontfamily{ppl}\fontseries{bx}}
   \renewcommand\gentryheader[4]{{\lettrine[lines=3,lhang=0.2,loversize=0.2]{\raisebox{0.1em}{#1}}{\textbf{\Large\textbf{~#4}}}}
   \ifstrempty{#1}{}{\linebreak\mbox{}~}}

On the last line, \mbox stops the \parindent space being ignored; the ~ brings it back up to the expected spacing.

### Package options

These options can be specified as a comma-separated list to the \usepackage line.

- **footnote** The footnote option causes \LaTeX to output footnotes at the end of each entry, or the end of each page, whichever comes first after the footnote mark.
   There are some limitations, described below.
   Support for other packages that affect footnotes is limited. If using hyperref, it is recommended to pass [hyperfootnotes=false] to avoid broken links.

- **noshuffle** In general, it’s easier to write gamebooks in a more linear fashion, in which related entries are kept together. But this is much less fun to play, as it’s too easy for the reader to simply read the adjacent entries to decide what to do. For this reason, this package shuffles the output entries by default. The first and last entries are never shuffled.
   Sometimes, perhaps for proofreading, you may not want the entries to be shuffled. In this case, you can use the noshuffle option.

- **verbose** This macro causes the package to output information messages about what it’s doing to the log file. This is not generally too useful, but it does include a mapping of the original paragraph indexes to their sorted positions, which may be useful to keep for handling proofreading corrections.

- **endpage** Puts the last entry on a page on its own. This typically produces a better “winning” feel for reaching the last entry, but it can also produce documents with ugly spacing, so it’s recommended to try it each way and see which works better for your text.
Footnotes

When typesetting a gamebook with footnotes, it is confusing if they migrate to the bottom of each page, as the footnote becomes visually detached from the entry to which it relates. Some effort has been taken to ensure that footnotes can be typeset at the bottom of the page on which the mark appears, or the bottom of each entry, whichever comes first.

However, this implementation has some limitations:

- Footnotes are not expanded until they are typeset.

- As a consequence, attaching one footnote to another with the use of \footnotemark within a \footnote argument will not advance the counter in time, so \footnotetext may not behave as you expect unless it is also set inside the first footnote’s text.

- Footnotes at the end of the page will not be broken across pages. Putting sufficient text into footnotes will cause the page to overflow.

- Where the footnote mark appears at the very end of a page, the footnote text may be set at the top of the subsequent page. \TeX{} is asked not to break the page there, but this influence is limited. It may be possible for the author to avoid this, e.g. by adding a rubber length to the interline spacing (it may be easier to simply choose a different seed value for the \texttt{lcg} package to reshuﬄe).

- If you have a lot of rubber space in the text, or variably-sized items, \TeX{} may expand the footnote at the end of the entry before it works out where to put the page break. In this case, the footnote will appear on the wrong page. The macro \texttt{\noentryfoot} is provided to allow the author to fix this case.

- Support for other footnotes packages may be limited and the behaviour of packages like \texttt{footnote}, \texttt{endnotes}, \texttt{footmisc}, \texttt{fnote}, \texttt{dblfnote} etc may be affected.

- If using the \texttt{hyperref} package, it is recommended to pass the option \texttt{hyperfootnotes=false} to that package, as the footnote links will be incorrect.

- The package uses \TeX\ \texttt{\mark}s to indicate which footnotes should appear on each page. Because \TeX{} supports only one set of marks, this would break any other package or usage of \texttt{\mark} while a gamebook was being output.

- Because \texttt{\mark} does not escape a floating environment, such as \texttt{minipage}, it is likely that this footnote implementation will not work as expected if the gamebook or \texttt{\footnote} is set in a \texttt{minipage} or similar.

- Each footnote is evaluated in a group. Just conceivably, this might affect the behaviour of some macros that affect the document beyond the footnote.
We rely on some non-“public” macros, such as `\@mpfn` and `\@footnotetext` defined by \LaTeX\ standard document classes. Other document classes may override these, which would override this package’s footnotes too.

For this reason, the improved footnote handling is only enabled if you pass the `footnote` option, and only while the environment is active.

**Implementation**

1 \ProvidesPackage{gamebooklib}[2021/08/24 Gamebook by R Lee latex@rjlee.homelinux.org]

We need \LaTeX\ 2ε, for the extra token registers.

2 \NeedsTeXFormat{LaTeX2e}[1994/06/01]

**verbose** The package option `verbose` enables detailed logging. Logging is via the macro `\gamebook@info`, which throws away detail messages unless the `verbose` option is given.

3 `\newcommand{\gamebook@info}[1]{}%`
4 `\DeclareOption{verbose}{%`
5 `\renewcommand{\gamebook@info}[1]{\PackageInfo{gamebooklib}{#1}}%`
6 `\gamebook@info{Gamebook Library package is logging}}%`

**endpage** The `endpage` option puts the last entry on its own page. This can work better when the last entry is about a page long, and also the final “winning” entry of the gamebook.

7 `\newcommand{\gamebook@beforelast}{}%`
8 `\DeclareOption{endpage}{%`
9 `\renewcommand{\gamebook@beforelast}{\eject}%`
10 `}%`

**footnote** The `footnote` option enables our footnote processing, to throw out footnotes at the end of each `gentry` so that they don’t appear to be against subsequent entries for that page.

This is generally recommended, unless you have a reason to turn it off (such as a conflicting package). It’s disabled by default because it could cause unexpected errors.

11 `\def\if@gamebook@footnotes{\iffalse}`
12 `\DeclareOption{footnote}{%`
13 `\gdef\if@gamebook@footnotes{\iftrue}%`
14 `\gamebook@info{Gamebook Library footnotes per gamebook entry}%`
15 `}%`

**noshuffle**

16 `\def\if@gamebook@shuffle{\iffalse}`
17 `\DeclareOption{noshuffle}{%`
18 `\gdef\if@gamebook@shuffle{\iftrue}%`
19 `\gamebook@info{Gamebook Library entries output in order}%`
20 `}%`
All unknown options are passed to \lcg, as it’s our only dependency with options. It’s suggested that users specify a value for “seed” for stable builds, by adding the following before including this package:
\usepackage[seed=123]{lcg}

\DeclareOption*{\PassOptionsToClass{\CurrentOption}{lcg}} \ProcessOptions\relax

We need to capture environment contents
\RequirePackage{environ}

\newcommand{\gentry}[1]{\@gentry{#1}}
\gentry just makes the first argument mandatory.

NB: If you define two entries with the same code, \LaTeX will print out a "multiply defined" label warning.

This can be used inside a `gentry`; it expands to the first optional argument of the `gentry` environment, which is either blank or the requested fixed index of the entry. To get the actual shuffled index, use \texttt{\gentryidxu}.

This can be used inside a `gentry`; it expands to the first mandatory argument of the `gentry` environment, which is the code for this entry (without the `gentry:` prefix).

\texttt{\@@gentry} then reads in the optional title argument, storing it in the \texttt{\gentrytitle} macro to supply the unsorted index number and the current entry’s code respectively.

For fixed entries, define a macro to hold the requested index

\texttt{\ifthenelse{\equal{\gentryidx}{}}{}{\expandafter\xdef\csname fixedat\arabic{gentryctr}\endcsname{\gentryidx}}}\%

\texttt{\expandafter\global\expandafter\newtoks\expandafter{\csname paratok\arabic{gentryctr}\endcsname}{}}}\%

This was supposed to discard any blank lines at the end of the `gentry` environment, but it still left odd spacing in the output somehow, and sometimes produced weird error messages about \texttt{\inaccessible} and \texttt{\head}. Removed for now.

\texttt{\newcommand{\gentry@store}{\edef{\head{\noexpand\begingroup{\noexpand\gentryheader{\arabic{gentryctr}}{\gentryidx}{\gentrycode}{\gentrytitle}{\noexpand\label{gentry:\gentrycode}}}}\noexpand\endgroup{}}}\%

Store the collected environment contents for \texttt{\thegentries} to output: This uses the token register \texttt{\gentry\texttt{N}}, where \texttt{N} is the unsorted index of this entry. Later, we’ll change the values of the \texttt{N} bit of \texttt{\gentry\texttt{N}} macros when we shuffle (the underlying token registers stay the same).

This relies on \texttt{\refstepcounter{\gentryctr}} before expanded before this macro. The label \texttt{\gentry:\gentrycode} is thus set to the current index of the final output entry.

\texttt{\newcommand{\gentry@store}{\edef{\head{\noexpand\begingroup{\noexpand\gentryheader{\arabic{gentryctr}}{\gentryidx}{\gentrycode}{\gentrytitle}{\noexpand\label{gentry:\gentrycode}}}}\noexpand\endgroup{}}}\%

Output the header, the environment token list, flush any footnotes (if applicable) then the inter-gentry footer.

\texttt{\expandafter{\noexpand\head{\texttt{\#}}}\%
\noexpand\flushfootnotes\texttt{\endgentry}\%
\gentryfooter{\noexpand\endgroup}}}\%
This macro is executed inside the gentry group and sets up the commands to be run to allow footnotes. It is only expanded if \if@gamebook@footnotes is true, i.e. the footnotes option is given.

65 \newcommand{\gentry@footnotespergentry}{}

\thegentries

This macro shuffles the entries as required, then expands to them in the correct order.

66 \newcommand{\thegentries}{%

This command is in a group so that the output routine resets. The expansion of \gentry@footnotespergentry only happens if the footnote option was given; it sets up the output routine while the gamebook is running.

67 \begingroup%
68 \if@gamebook@footnotes\gentry@footnotespergentry\fi%

To begin, let’s record the number of entries. This may come in useful. Note that you can use this macro in the gentry environment, because that’s not been expanded yet.

69 \xdef{\gentrycount}{\arabic{\gentryctr}}%

The next thing is to perform some surgery on LCG. This cuts out an annoying warning, hopefully more reliably than replacing the definition of \p@stkeysr@nd. The warning is simply that this package doesn’t waste another counter every time it changes the random limits (which happens a lot during Knuth shuffle):

70 \WarningFilter{lcg}{Using an already existing counter rand}%

The output routine and end-of-page footnotes

The idea is to keep footnotes always on the same page as their mark where possible.

\TeX does lots of fun things with the output routine, which we want to keep. So grab a copy of whatever the code is currently doing:

Here I’m using the \edef trick to expand \the\output into a token register, because using a macro causes a weird error about an “{” after “\the”.

71 \newtoks{\gentry@oldoutput}{%
72 \edef{\mytmp}{\noexpand{\gentry@oldoutput={\the\output}}}

For the footnotes, if we reach the end of a page without outputting them, we need to flush them.

\output is the output routine. It takes the page built up in \box255, annotates it with headers, footers etc, then ships it out. For our purposes, we only need to append the footnotes to the bottom of \box255.

73 \if@gamebook@footnotes\output{%

The \outputpenalty tells us why the output routine was called; generally, it’s invoked whenever a new floatable environment is generated, or when a page is full. Anything less than -1000 means that the page was filled, so we should add any footnotes only in this case.
The Shuffling Algorithm

The basic shuffling algorithm is to first shuffle all entries, except for those marked with a fixed index, then to go through the fixed-index entries in order and swap them into their final place.

The original version of this package had a bug relating to multiple fixed-index entries. In short, let $A$, $B$, and $C$ be indices; if $A < B$ and (unshuffled) entry number $A$ was fixed at (shuffled) location $B$, while (unshuffled) entry number $B$ was fixed at (shuffled) location $C$, so during the “shuffle,” $A$ would be swapped with $B$, then $B$ would be swapped with $C$, resulting in $A$ appearing at $C$ in the text, not $B$ as requested. Because the unshuffled index doesn’t appear in either the source or output document, this could be difficult to diagnose; the author simply saw one of their entries ending up in the wrong place.

TEX has well beyond 255 token registers these days, so don’t bother to check that limit.

The LCG package provides a suitable pseudo-random number generator. What we want is a repeatable series of dispirate numbers, not an especially random one.

1. Work out how many entries there are ($N$). Provided \texttt{the\textgencnties} is called at the end, this is just the value of \texttt{gentryctr}

2. Declare a set of token registers named \texttt{paratoks}$_n$, where $n$ is each integer 1...$N$ inclusive. These will hold the contents of the entry.

3. Declare a set of macros named \texttt{paraidx}$_n$, where $n$ is each integer 1...$N$ inclusive, each initialised to $n$. These will hold the number of the entry.

4. Shuffle elements \{2 : $N - 1$\}, in that array. For $i = 2$ through $N - 2$
   
   (a) Let $R$ be a random number between $i$ and $N - 1$ inclusive
   (b) If $R \neq i$ then swap macros \texttt{gentrytoks}$_R$ and \texttt{gentrytoks}$_i$
   (c) If $R \neq i$ then swap macros \texttt{paraidx}$_R$ and \texttt{gentryidx}$_i$

5. For $i = 1 : n$, output token reg $i$

Define macros \texttt{\csname paraidx\textendcsname} containing the arabic original page number to be put out on the $n$th output “page”

\xdef\gentrycount{\arabic{gentryctr}}

9
we can reuse \texttt{gentryctr}; we’ve finished this set of paras and kept the total count.

\begin{verbatim}
\setcounter{gentryctr}{0}
\ifthenelse{\gentrycount<3}{% SHUFFLE START
  \whiledo{\not{\value{gentryctr}>=\gentrycount}}{%
    \edef\gentryidxu{\arabic{gentryctr}}%
    \expandafter\xdef\csname paraIdx\gentryidxu\endcsname{\gentryidxu}%
    \stepcounter{gentryctr}%
  }%

\textbf{gentryidxu} The \texttt{gentryidxu} macro can be used inside a \texttt{gentry} to obtain the current arabic shuffled index of the entry.

\textbf{gentryidxs} The \texttt{gentryidxs} macro can be used inside a \texttt{gentry} to obtain the arabic original unshuffled index of the entry (the first \texttt{gentry} is 1, and this counter resets after each expansion of \texttt{\thestep\theentries}).

\begin{verbatim}
\if\@gamebook\@shuffle
  \setcounter{rand}{\gentrycount}%
  \addtocounter{rand}{-1}\edef\stoppoint{\arabic{rand}}%
  \setcounter{gentryctr}{2}%
  \chgrand[last=\stoppoint]%
  \whiledo{\value{gentryctr}<\stoppoint}{%
    \edef\gentryidxu{\arabic{gentryctr}}%
    \expandafter\ifcsname fixedat\gentryidxu\endcsname%
      \gamebook@info{Not shuffling \gentryidxu; fixed pos}%
      \stepcounter{gentryctr}%
    \else%
      \gamebook@info{Shuffling \gentryidxu}%
      \stepcounter{gentryctr}%
      \edef\nextidx{\arabic{gentryctr}}%
      \chgrand[first=\nextidx]\rand%
      \expandafter\ifcsname fixedat\arabic{rand}\endcsname\else%
        \gamebook@info{Shuffling \gentryidxu to \arabic{rand}}%
        \macroswap{paraIdx\gentryidxu}{paraIdx\arabic{rand}}%
      \fi%
    \fi%
  }
\end{verbatim}

First, shuffle everything that isn’t fixed down. Don’t renumber para 1 or \texttt{\gentrycount}; Knuth-shuffle the rest NB: we stop at \texttt{\gentrycount} – 2, because \texttt{\gentrycount} – 1 would only shuffle with itself.

\begin{verbatim}
\setcounter{gentryctr}{2}%
\chgrand[last=\stoppoint]%
\whiledo{\value{gentryctr}<\stoppoint}{%
  \edef\gentryidxu{\arabic{gentryctr}}%
  \expandafter\ifcsname fixedat\gentryidxu\endcsname%
    \gamebook@info{Not shuffling \gentryidxu; fixed pos}%
    \stepcounter{gentryctr}%
  \else%
    \gamebook@info{Shuffling \gentryidxu}%
    \stepcounter{gentryctr}%
    \edef\nextidx{\arabic{gentryctr}}%
    \chgrand[first=\nextidx]\rand%
    \expandafter\ifcsname fixedat\arabic{rand}\endcsname\else%
      \gamebook@info{Shuffling \gentryidxu to \arabic{rand}}%
      \macroswap{paraIdx\gentryidxu}{paraIdx\arabic{rand}}%
    \fi%
  \fi%
}\end{verbatim}

If this is to be swapped with a fixed position, skip it

Roll the dice. If we’ve hit an entry with fixed position, we must skip it, or it would end up being swapped out into \texttt{fixedat\arabic{rand}} instead.
Now move fixed entries into their final place:

\setcounter{gentryctr}{2}\%
\whiledo{\not{\value{gentryctr}>=\stoppoint}}{\%
  \edef\gentryidxu{\arabic{gentryctr}}\%
  \expandafter\ifcsname fixedat\gentryidxu\endcsname\%
  \expandafter\edef\expandafter\mydest\expandafter{\expandafter\csname fixedat\gentryidxu\endcsname}\%
  \gamebook@info{MOVING FIXED GAMEBOOK ENTRY INTO PLACE: \gentryidxu -> \mydest}\%
  \macroswap{paraIdx\gentryidxu}{paraIdx\expandafter\csname fixedat\gentryidxu\endcsname}\%
  \gamebook@info{MOVING FIXED GAMEBOOK ENTRY INTO PLACE: \gentryidxu -> \mydest}\%
  \expandafter\ifcsname fixedat\mydest\endcsname\%
  \macroswap{\expandafter\csname fixedat\gentryidxu\endcsname}{\fixedat\mydest}\%
  \global\let\csname fixedat\mydest\endcsname\@undefined\%
  \addtocounter{gentryctr}{-1}\%
\fi\%
\fi\%
\stepcounter{gentryctr}\%
}\fi\shufflee nd\%
\gamebook@info{Shuffled! Gentry order:}\%
\setcounter{gentryctr}{1}\%
\whiledo{\not{\value{gentryctr}>=\gentrycount}}{\%
  \edef\gentryidxu{\arabic{gentryctr}}\%
  \gamebook@info{\gentryidxu -> \csname paraIdx\gentryidxu\endcsname}\%
  \refstepcounter{gentryctr}\%
  \ifthenelse{\value{gentryctr}=\gentrycount}{\%
    \gamebook@beforelast\%
  }{}\%
  \edef\gentryidxs{\csname paraIdx\gentryidxu\endcsname}\%
  \gamebook@info{Output gentry \gentryidxu\ of \gentrycount, original idx \gentryidxs}\%
  \the\csname paratok\gentryidxs\endcsname\%
}\%

We can reuse the same counter again to output
\setcounter{gentryctr}{0}\%
\gamebook@info{Outputting \gentrycount\ gamebook entries}\%
\whiledo{\value{gentryctr}<\gentrycount}{\%
  \refstepcounter{gentryctr}\%
  \ifthenelse{\value{gentryctr}=\gentrycount}{\%
    \gamebook@beforelast\%
  }{}\%
  \edef\gentryidxu{\arabic{gentryctr}}\%
  \edef\gentryidxs{\csname paraIdx\gentryidxu\endcsname}\%
  \gamebook@info{Output gentry \gentryidxu\ of \gentrycount, original idx \gentryidxs}\%
  \the\csname paratok\gentryidxu\endcsname\%
}\%
Finally, we clear the registers and reset the counter in case we want to start again.

```
\setcounter{gentryctr}{1}%
\whiledo{\not{\value{gentryctr} > \gentrycount}}{%
  \edef\gentryidxu{\arabic{gentryctr}}%
  \expandafter\global\expandafter\let\csname paratok\gentryidxu\endcsname@undefined%
  \expandafter\ifcsname fixedat\gentryidxu\endcsname%
    \expandafter\global\expandafter\let\csname fixedat\gentryidxu\endcsname@undefined%
  \fi%
  \stepcounter{gentryctr}%
}\setcounter{gentryctr}{0}%
eject
\endgroup%
```

The final `\eject` ensures that the output routine has flushed as many pages as it can, before the output routine is reset again.

```
\gentryheader
\gentryfooter
```

This macro is called before outputting the header. Its job is to format whatever header information the user wants to see on each entry; generally, this will be the page number.

It takes 4 parameters:

1. The `unshuffled` index value (to help an author find an entry in the original text)
2. The `fixed` index value, if any; otherwise an empty argument
3. The user-entered unique code for this entry
4. The user-supplied title, if any; otherwise an empty argument

The value of the output index can be obtained with `\arabic{gentryctr}`

```
\newcommand{\gentryheader}{4}%
\noindent\textbf{\Huge\arabic{gentryctr}\large\ #4}%
\nopagebreak%
\vspace{0.3em}%
\nopagebreak%
\par%
\marginpar{\#3}%
```

This takes no arguments and is simply expanded after the entry is typeset. The default adds some vertical space and a simple separator.

```
\newcommand{\gentryfooter}{}%\par\vspace{2em}\centerline{---}\vspace{2em plus 1in}\par%
```

12
This is a technique described on StackExchange:
https://tex.stackexchange.com/questions/179016/
ignore-spaces-and-pars-after-an-environment#179034

This is used to ensure that extra space at the start and end of the `gentry`
environment is ignored.

```latex
\def\ignorespacesandallpars@ {% 
  \@ifnextchar\par{% 
    \expandafter\ignorespacesandallpars@gobble} {} 
}
```

\footnotetext Token registers to hold each footnote.

Each footnote is held in its own token register, so that we can control which
footnotes appear on each page.

\TeX would normally use saveboxes to store footnotes, but I prefer to hold off
on expanding them until we know which page they’re to be expanded on, which
saves some difficulties (and perhaps creates others).

To start with, modify \TeX’s `\makefnmark` to output a mark for the footnote,
holding its index. This tells the output routine which footnotes to include at the
end of the page. The `\in@out` macro is only defined when outputting footnotes,
so suppresses this mark when we don’t need it.

```latex
\g@addto@macro{\@makefnmark}{% 
  \ifcsname in@out\endcsname\else% 
    \mark{\arabic{\@mpfn}}% 
  \fi% 
}
```

The `\footnotetext@save` is a convenience that takes the the footnote counter
value in the first argument, the token register as a second argument, and the token
list (footnote text) as the third.

```latex
\g@addto@macro{\@footnotetext@save}{% 
  \newcommand{\@footnotetext@save@}[2]{% 
    \edef\@tmp@{\expandafter\arabic{\@mpfn}}% 
    \expandafter\@footnotetext@save\expandafter{\@tmp@}{#1}{#2}% 
  }% 
}
```

Footnotes are built in a group, so that `\in@out` can be defined locally to indicate
that footnotes are being built (which stops spurious `\marks`).

```latex
\g@addto@macro{\gentry@footnotespergentry}{% 
  \@footnotetext@save@} 
```
\renewcommand{\@footnotetext}{%}
\begingroup
\def\in@out{}% flag that we’re building footnotes
\edef\@tmp{\expandafter\csname footnotetoks\arabic{\@mpfn}\endcsname}%
\expandafter\@footnotetext@save@\expandafter{\@tmp}{%}
% This fixes up the use of \cs{footnotemark} within footnotes:
#1%
\endgroup%
}}%
\begin{thebibliography}{9}
\footnotesize
\setlength{\itemsep}{0pt}
\end{thebibliography}
\end{document}
First, we check if the macro exists. If it does, it must contain the value of the
token register.
Next, we make sure that the token register has contents.
\expandafter\ifcsname footnotetoks\arabic{fncounter}\endcsname%
\edef\tmp@@\csname footnotetoks\arabic{fncounter}\endcsname%
This \texttt{detokenize} black magic tests if a token reg is actually empty:
https://tex.stackexchange.com/questions/263733/
whats-the-best-practice-way-to-test-whether-parameter-is-empty
\expandafter\if\expandafter\relax\expandafter%
\texttt{detokenize}\expandafter{\the\tmp@@}\relax\else%
\footnote@rule\gdef\footnote@rule{\ifvmode\else\par\fi}%
\interlinepenalty\interfootnotelinepenalty%
Actually output the footnotes.
Having output the footnote, clear the token register (to save memory), then
use \texttt{let} to clear the definition of the macro. This ensures that we don’t try to
output the same footnote twice (at page end and entry end).
\expandafter\the\tmp@@%
\global\expandafter\tmp@@={}%
\expandafter\let\tmp@@\@undefined%
\stepcounter{fncounter}%
}\endgroup%
\LaTeX{} works by building up a little more than a page, then calling the output
routine. The output routine then decides where to put the page end from the
built-up material. If there is a footnote mark, it could come after the page end, so
we can’t rely on the fact that there’s footnote register to determine if we should
output footnotes or not. The edge case is: if the footnote mark is held back for
the next page, the footnote text would appear on the footer of the page where
being built when the mark was expanded, which is the page before the footnote.
One fix for this is to have each footnote mark in the main body of the text
output a \texttt{\mark} containing the footnote’s counter value, and output the footnotes
only to \texttt{\botmark}, which is the last \texttt{\mark} actually typeset on the page.
(see https://www.tug.org/TUGboat/Articles/tb11-4/tb30salomon.pdf, page
598, to read around footnotes).
This macro is called at the end of each \texttt{gentry}. It switches into vertical mode
(as ending \texttt{gentry} doesn’t throw a \texttt{\break} or line end by itself), then we output all
footnotes so far (by reading the footnote counter \texttt{\@mpfn}, as this is “sequential”).
\newcommand{\outputfootnotes@endgentry}{%
\if@gamebook@footnotes%
\nopagebreak\ifhmode\fi% get into vertical mode
\nopagebreak\ifmode\fi% get into vertical mode
\outputfootnotes{\arabic{\@mpfn}}%
\fi%
}%
\outputfootnotes@endgentry
This method simply suppresses footnotes at the end of the entry within the current
group. This forces footnotes within the current group to be printed at the bottom
of the page.

This is useful in the case where \LaTeX{} expands the footnote at the end of the
entry, then decides to put the page split between the footnote mark and the text.

\begin{verbatim}
\newcommand{\noentryfoot}{\def\outputfootnotes@endgentry{}}
\end{verbatim}

This is called by the output routine at the end of each page. If \texttt{\botmark} has a
value, then we can output all footnotes up to that index.

\begin{verbatim}
\g@addto@macro{\gentry@footnotespergentry}{
\newcommand{\outputfootnotes@endpage}{%
\expandafter\if\expandafter\relax\expandafter\detokenize{\botmark}\relax\else%
\outputfootnotes{\botmark}\%
\fi%
}}%
\end{verbatim}

\section*{Change History}

\begin{tabular}{lll}
\textbf{v1.0} & \textbf{v1.1} & \\
General: Initial Release & General: Bugfix: edge case with
& \\
endpage: Endpage option & clashing fixed-index entries & 1
& \\
footnote: Footnote option & gentryidxs: Access original index & 10
& \\
noshuffle: Noshuffle option & gentryidxu: Access shuffled index & 10
& \\
verbose: Verbose option & \\
\end{tabular}
Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols
\@gentry ........ 36, 43
@footnotemark .... 191
@footnotetext . 13, 199
@footnotetext@save ........ 189, 196
@footnotetext@save@ ........ 194, 203
@gentry ........ 32
@makefnmark ..... 183
@mpfn 185, 195, 202, 240
\_ ........ 139, 147, 168
C 96, 106
chgrand ....... 96, 106
E 9, 164
\eject .... 9, 164
\endpage .... 1
\n environments:
gentry ........ 32
F
@footnote .... 1, 11
@footnote@rule 213, 226
G 7, 9, 143
\gamebook@beforelast ........ 3, 5, 6, 14, 19, 100, 103, 109, 120, 131, 135, 139, 147, 151
\gamebook@info 164
\ifgamebook@shuffle .... 16, 18, 91
\gentry ........ 32
@gamebook@shuffle ........ 37, 53, 180
\gentry@footnotespergentry\ignorespacesandallpars .... 8, 65, 68, 188, 193, 198, 244
\gentry@oldoutput .... 71, 72, 81
@gentry@store .... 52, 55
@gamebook@info 3, 5, 6, 11, 13, 68, 73, 238
@macroswap 110, 121, 124
@mydest 118, 120, 123–125
\nentryfoot .... 16, 243
\nshuffle .... 1, 16
N
nextidx .... 105, 106

O
\output .... 72, 73
\outputfootnotes .... 14, 209, 240, 248
\outputfootnotes@endgentry .... 15, 62, 237, 243
\outputfootnotes@endpage .... 16, 78, 245
\stoppoint 94, 96, 97, 115

S
\theentries .... 8, 66
\ifgamebook@footnotes 11, 13, 68, 73, 238
\verbose .... 1, 2