The Gamebooklib Package

Robert J Lee
latex@rjlee.homelimux.org
August 19, 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\textsuperscript{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.

\textsuperscript{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:

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

Turn to `\turnto{codeb}`
```
\begin{gentry}
\[123\]{codeb}[title]
This is the second and final entry.\par
To play again, turn to `\turnto{codea}`
\end{gentry}

\% Output:
\thegentries
```

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.

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{\raisebox{0.3em}{~#4}
   \ifstrempty{#1}{}{\linebreak\mbox{}~}}%
   \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.

3
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, eg 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 `\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 `footnote`, `endnotes`, `footmisx`, `fnote`, `dblfnote` etc may be affected.
- If using the `hyperref` package, it is recommended to pass the option `hyperfootnotes=false` to that package, as the footnote links will be incorrect.
- The package uses \TeX{} `\marks` 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 `\mark` while a gamebook was being output.
- Because `\mark` does not escape a floating environment, such as `minipage`, it is likely that this footnote implementation will not work as expected if the gamebook or `\footnote` is set in a `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 \texttt{footnote} option, and only while the environment is active.

\section*{Implementation}

1 \ProvidesPackage{gamebooklib}[2021/08/10 Gamebook by R Lee latex@rjlee.homelinux.org]

We need \LaTeX{} 2\epsilon, for the extra token registers.

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

\texttt{verbose} The package option \texttt{verbose} enables detailed logging. Logging is via the macro \verb|gamebook@info|, which throws away detail messages unless the \texttt{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}%

\texttt{endpage} The \texttt{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 }

\texttt{footnote} The \texttt{footnote} option enables our footnote processing, to throw out footnotes at the end of each \texttt{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 }

\texttt{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}

21 \DeclareOption*{%
22 \typeout{Current option: \CurrentOption}%
23 \PassOptionsToClass{\CurrentOption}{lcg}%
24 }%
25 \ProcessOptions\relax%

We need to capture environment contents
26 \RequirePackage{environ}%

Macroswap: used to swap the commands that evaluate the macros
27 \RequirePackage{macroswap}%

Ifthen makes branching and loops a little easier
28 \RequirePackage{ifthen}

LCG: random numbers for the shuffle
29 \RequirePackage{lcg}%

Silence is used to suppress a warning from `lcg` that gamebook is not wasting counter registers. `debrief` ensures that the user is at least told than warnings were suppressed.

To see the warnings, put the following line before \usepackage{gamebook}:

\usepackage[debrief,showwarnings]{silence}:

30 \RequirePackage[debrief]{silence}%

Let’s count the entries we’re reading in so we can build up token registers.
31 \newcounter{gentryctr}%
32 \setcounter{gentryctr}{0}%

`gentry` \{(\textit{fixedIdx})\}{\{\textit{code}\}}{\{\textit{title}\}} First, we need to parse the optional arguments.

When we say \begin{\textit{gentry}}, \LaTeX{} does some stuff ending in \textit{gentry} so we redefine \textit{gentry} to take an optional argument and delegate to \texttt{@gentry}
33 \newcommand{\gentry}[1]{\@gentry{#1}}%

\gentry just makes the first argument mandatory.

\@gentry defines \texttt{gentryidx} and \texttt{gentrycode}, which can be used in the body of the \textit{gentry} environment to supply the unsorted index number and the current entry’s unique code.

NB: If you define two entries with the same code, \LaTeX{} will print out a ”multiply defined” label warning.
34 \newenvironment{\textit{gentry}}[2]{%
35 \def\gentryidx{#1}%
36 \def\gentrycode{#2}%
37 \@gentry%
38 }{%
39 ignorespacesandsallpars%
40 }%
\@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.

\begin{verbatim}
\newcommand{\@@gentry}[1]{%\def\gentrytitle{#1}\stepcounter{gentryctr}}
\end{verbatim}

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

\begin{verbatim}
\ifthenelse{\equal{\gentryidx}{}}{}{\expandafter\xdef\csname fixedat\arabic{gentryctr}\endcsname{\gentryidx}}\expandafter\global\expandafter\newtoks\expandafter{\csname paratok\arabic{gentryctr}\endcsname}{ }\Collect@Body\gentry@store%
\end{verbatim}

This was supposed to discard any blank lines at the end of the \texttt{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.

\begin{verbatim}
\% \ignorespacesandallpars%
\}
\end{verbatim}

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

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

\begin{verbatim}
\newcommand{\gentry@store}[1]{%\edef\head{\noexpand\begingroup\noexpand\gentryheader\arabic{gentryctr}\gentryidx\gentrycode\gentrytitle\noexpand\label{gentry:\gentrycode}\noexpand\endgroup}\global\expandafter\csname paratok\arabic{gentryctr}\endcsname=}\Collect@Body\gentry@store%
\end{verbatim}

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

\begin{verbatim}
\expandafter{\head \arabic{gentryctr}\outputfootnotes\endgentry\gentryfooter\endgroup}%
\end{verbatim}

\texttt{\gentry@footnotespergentry} This macro is executed inside the \texttt{gentry} group and sets up the commands to be run to allow footnotes. It is only expanded if \texttt{\textbackslash if@gamebook\textbackslash footnotes} is true, ie the \texttt{footnotes} option is given.

\begin{verbatim}
\newcommand{\gentry@footnotespergentry}{}
\end{verbatim}

\texttt{\thegentries}
This macro shuffles the entries as required, then expands to them in the correct order.

\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.

\begingroup%
\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.

\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):

\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. \LaTeX{} 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”.

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

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.

\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.
\ifnum\outputpenalty<-1000\else%
\edef\tmp@{footnotetoks\botmark}
\expandafter\ifcsname footnotetoks\botmark\endcsname%
\global\setbox255=\vbox to \vsize{%
\unvbox255\vfill\outputfootnotes@endpage}%
\fi%
\fi%
The shuffling algorithm

\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 numbers.

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

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

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

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

5. For \textit{i} = 1 : \texttt{n}, output token reg \textit{i}

Define macros \texttt{\csname paraidx\textit{N}'\endcsname} containing the arabic original page number to be put out on the \textit{N}'th output “page”

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

If there are fewer than 3 entries, don’t try and shuffle.

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

\edef\myidx{\arabic{gentryctr}}%
\expandafter\ifcsname fixedat\myidx\endcsname%
\gamebook@info{Not shuffling \myidx; fixed pos}%
\stepcounter{gentryctr}%
\else%
\gamebook@info{Shuffling \myidx}%
\stepcounter{gentryctr}%
\edef\nextidx{\arabic{gentryctr}}%
\fi%

If we've hit an entry with fixed position, we may as well skip it
\expandafter\ifcsname fixedto\arabic{rand}\endcsname\else%
\chgrand[first=\nextidx]
\rand%
\gamebook@info{Shuffling \myidx to \arabic{rand}}%
\macroswap{paraIdx\myidx}{paraIdx\arabic{rand}}%
\fi%
\fi%

Now move fixed entries into their final place:
\setcounter{gentryctr}{2}%
\whiledo{\value{gentryctr}<\stoppoint}{%
\edef\myidx{\arabic{gentryctr}}%
\expandafter\ifcsname fixedat\myidx\endcsname%
\expandafter\edef\expandafter\mydest\expandafter{\expandafter\csname fixedat\myidx\endcsname}%
\gamebook@info{MOVING FIXED GAMEBOOK ENTRY INTO PLACE: \myidx -> \mydest}%
\macroswap{paraIdx\myidx}{paraIdx\mydest}%
\fi%
\stepcounter{gentryctr}%
\}SHUFFLE END

We can reuse the same counter again to output
\setcounter{gentryctr}{0}%
\gamebook@info{Outputting \gentrycount\ gamebook entries}%
\whiledo{\value{gentryctr}<\gentrycount}{%
\stepcounter{gentryctr}%
\ifthenelse{\value{gentryctr}=\gentrycount}{%
The final \texttt{\textbackslash eject} ensures that the output routine has flushed as many pages as it can, before the output routine is reset again.  

\gentryheader

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 \texttt{unshuffled} index value (to help an author find an entry in the original text)
2. The \texttt{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 \texttt{\arabic{gentryctr}}.
\g@addto@macro\@makefnmark{% 
\ifcsname in@out\endcsname\else% 
\mark{\arabic{\@mpfn}}% 
\fi% 
}%

\g@addto@macro\@footnotetext{% 
\global\newtoks#2{}% 
\global#2={\noindent\@footnotemark{}#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%
}%

This is a technique described on StackExchange:
https://tex.stackexchange.com/questions/179016/
ignore-spaces-and-par

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

\def\ignorepar{% 
\@ifnextchar\par{% 
\expandafter\ignorepar\@gobble}%
}{}

\gentryfooter

\ignorespacesandallpars@

This is a technique described on StackExchange:
ignore-spaces-and-par-after-an-environment#179034

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

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

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

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

\ignorespacesandallpars@

\@footnotetext@save (token register) Token registers to hold each footnote.

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

\LaTeX 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 \LaTeX’s \MakeFntmark 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.

\g@addto@macro\MakeFntmark{% 
\ifsname in@out\endsname\else% 
\mark{\arabic{\@mpfn}}% 
\fi% 
}%

The \footnotetext@save (token register) 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.

\g@addto@macro\gentry\footnotespergentry{% 
\neucmd\@footnotetext@save[3]{% 
\global\newtoks#2\@gobble}:% 
\global\noindent\@footnotemark{}#3% 
}%

\@footnotetext@save unpacks the arguments for \@footnotetext@save. This one uses the value of the \@mpfn counter to unpack the footnote counter; the token
register (full csname) as the first argument, and the token list (footnote text) as the second.

\g@addto@macro{\gentry@footnotespergentry}{% 
\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).
\g@addto@macro{\gentry@footnotespergentry}{% 
\renewcommand{\@footnotetext}{1}{% 
\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%}
\endgroup
}}%

\outputfootnotes {\langle maxIdx \rangle}

Command to output the footnotes, which are just in \footnotetoks\ for now. An end user can call this at any point in the text to set out the footnotes.

This macro takes one argument, being the maximum index of footnotes to output, inclusive. Footnotes after this index will be excluded. If not provided, the value provided by the counter \@mfpn, which in the default \LaTeX counter, will be used.
\newcounter{fncounter}%%
\newcommand{\outputfootnotes}[1]{% 
\begingroup% 
\def\in@out{}% flag that footnotes are outputting; suppresses marks 
\setcounter{fncounter}{1}%%
\begin{group}%
\def\footnote@rule{\langle maxIdx \rangle}

The next line comes from the TUGboat suggestions, and protect against various user changes
\leftskip=0pt\rightskip=0pt\interlinepenalty=1000%%
\penalty-1000% 
\vspace{1pt plus 2pt minus 0.5pt}%%
\hspace{-0.5in}\rule{1.5in}{0.4pt}%%
\end{group}%

Called in vertical mode, and don’t want to throw a page break.

I’m not sure how \LaTeX renders the footnote rule exactly, so I’m just using my own.
\def\footnote@rule{\langle maxIdx \rangle}

Invoke the output routine. This attempts to stop the output routine being invoked while we’re adding the footnote rule, which could cause a blank footnote to appear.

Instead, it means some footnotes could appear on the subsequent page. To guard against this, footnotes are set into a \vbox to prevent page breaking.
The macro \footnote@rule is output before each footnote. The first time, it outputs a divider rule; subsequently, it just throws a new paragraph to keep footnotes on separate lines.

The next line comes from the TUGboat suggestions, and protect against various user changes
\footnote@rule
\vbox{%
\whiledo{\not{\value{fncounter}>#1}}{%
We are now looping over all possible entry numbers, in order. Some will already have been output, but we check them all anyway (it doesn’t take much time).

First, we check if the macro exists. If it does, it must contain the value of the token register.
\expandafter\ifcsname footnotetoks\arabic{fncounter}\endcsname%
\edef\tmp@@{\csname footnotetoks\arabic{fncounter}\endcsname}%
This \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%\detokenize\expandafter{\the\tmp@@}\relax\else%
\footnote@rule\gdef\footnote@rule{\ifvmode\else\par\fi}%
Actually output the footnotes.

Having output the footnote, clear the token register (to save memory), then use \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\expandafter\tmp@@\@undefined%
\fi\fi%
\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 \mark containing the footnote’s counter value, and output the footnotes only to \botmark, which is the last \mark actually typeset on the page.
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”).

\begin{verbatim}
\newcommand{\outputfootnotes@endgentry}{\if@gamebook@footnotes\nopagebreak\ifhmode\fi% get into vertical mode \nopagebreak\outputfootnotes{\arabic{\@mpfn}}\fi% \}%
\end{verbatim}

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{\expandafter{\botmark}}}{\relax}{\else}{\outputfootnotes{\botmark}}\relax{\else}}\fi%}}}
\end{verbatim}

\section{Change History}

\begin{tabular}{ll}
\texttt{v1.0} & \texttt{footnote}: Footnote option \ldots \ldots \ 3 \\
General: Initial Release \ldots \ldots \ldots \ldots \ 1 & \texttt{noshuffle}: Noshuffle option \ldots \ldots \ 3 \\
\texttt{endpage}: Endpage option \ldots \ldots \ldots \ldots \ 3 & \texttt{verbose}: Verbose option \ldots \ldots \ 3 \\
\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 37, 40
\@footnotemark 183
\@footnotetext 12, 191
\@footnotetext@save 181, 188
\@footnotetext@save@ 186, 195
\@gentry 33
\@makefnmark 175
\@mpfn 177, 187, 194, 232
\@ 131, 139, 160
C
\chgrand 93, 104
E
\eject 9, 156
\endpage 1 2
environments:
gentry 33
F
\footnote 1, 11
\footnoterule 205, 218
G
\gamebook@beforelast 7, 9, 135
\gamebook@info 3, 5, 6
\gamebook@info 14, 19, 97, 100
\gamebook@info 106, 117, 123
\gamebook@info 127, 131, 139, 143
\gamebook@info 33
\gamebook@info (environment) 33
\gamebook@footnotespergentry 7, 62, 65, 180, 185, 190, 236
\gamebook@footnotespergentry 68, 69, 78
\gamebook@store 49, 52
\gamebookcode 36, 54, 55
\gamebookcount 66, 80, 82, 83, 90, 125, 131, 132, 134, 139, 145
\gamebookfooter 12, 60, 167
\gamebookheader 2, 11, 53, 159
\gentryidx 35, 43, 45, 54
\gentrytitle 41, 54
\in\out 193, 203
M
\macroswap 107, 118
\mydest 115, 117
N
\nextidx 102, 104
\noentryfoot 15, 235
\noshuffle 1, 16
O
\output 69, 70
\outputfootnotes 13, 201, 232, 240
\outputfootnotes@endgentry 15, 59, 229, 235
\outputfootnotes@endpage 15, 75, 237
S
\stoppoint 92-94, 112
T
\thegentries 7, 63
\if@gamebook@footnotes 11, 13, 65, 70, 230
V
\if@gamebook@shuffle 16, 18, 88
\ignorespacesandallpars 38, 50, 172
\ignorespacesandallpars@ 12, 170
Y
\youridx 138, 140, 141