

# The Gamebooklib Package<sup>\*</sup>

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 L<sup>A</sup>T<sub>E</sub>X 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<sup>1</sup>. 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.

---

<sup>\*</sup>This document corresponds to Gamebooklib Gamebook, dated 2021/08/24.

<sup>1</sup><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

Turn to \turnto{codeb}
\end{gentry}

\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.

`\gentryheader`     `{\langle counterIdx \rangle}{\langle fixedIdx \rangle}{\langle code \rangle}{\langle title \rangle}`

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}}}}%
  }%
  \ifstrepty{#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 L<sup>A</sup>T<sub>E</sub>X 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, eg by adding a rubber length to the interline spacing (it may be easier to simply choose a different seed value for the `lcg` package to reshuffle).
- If you have a lot of rubber space in the text, or variably-sized items, `LATEX` 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`, `footmisc`, `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 L<sup>A</sup>T<sub>E</sub>X 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{\iftrue}
17 \DeclareOption{noshuffle}{%
18   \gdef\if@gamebook@shuffle{\iffalse}%
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   \PassOptionsToClass{\CurrentOption}{lcg}%
23 }%
24 \ProcessOptions\relax%
```

We need to capture environment contents

```
25 \RequirePackage{environ}%
```

Macroswap: used to swap the commands that evaluate the macros

```
26 \RequirePackage{macroswap}%
```

Ifthen makes branching and loops a little easier

```
27 \RequirePackage{ifthen}
```

LCG: random numbers for the shuffle

```
28 \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}:
```

```
29 \RequirePackage[debrief]{silence}%
```

Let's count the entries we're reading in so we can build up token registers.

```
30 \newcounter{gentryctr}%
31 \setcounter{gentryctr}{0}%
```

**gentry** [*fixedIdx*][*code*][*title*] First, we need to parse the optional arguments.

When we say `\begin{gentry}`, L<sup>A</sup>T<sub>E</sub>X does some stuff ending in `\gentry` so we redefine `\gentry` to take an optional argument and delegate to `\@gentry`

```
32 \newcommand{\gentry}[1][]{\@gentry{#1}}%
```

`\gentry` just makes the first argument mandatory.

NB: If you define two entries with the same code, L<sup>A</sup>T<sub>E</sub>X will print out a "multiply defined" label warning.

```
33 \newenvironment{@gentry}[2]{%
34   \xdef\gentryidx{#1}%
35   \xdef\gentrycode{#2}%
36   \@gentry%
37 }{\ignorespacesandallpars%
38   \global\let\gentryidx\@undefined%
39   \global\let\gentrycode\@undefined%
40   \global\let\gentryidxu\@undefined%
41   \global\let\gentryidxs\@undefined%
42 }%
```

**gentryidx** 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 **\gentryidxu**.

**gentrycode** 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).

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

```
43 \newcommand{\@@gentry}[1][]{%
44   \def\gentrytitle{#1}%
45   \stepcounter{gentryctr}%
```

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

```
46   \ifthenelse{\equal{\gentryidx}{}}{}{%
47     \expandafter\xdef\csname fixedat\arabic{gentryctr}%
48     \endcsname{\gentryidx}%
49   }%
50   \expandafter\global\expandafter\newtoks\expandafter{%
51     \csname paratok\arabic{gentryctr}\endcsname}{ }%
52   \Collect@Body\gentry@store%
```

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 **\inaccessible** and **\head**. Removed for now.

```
53 % \ignorespacesandallpars%
54 }
```

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

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

```
55 \newcommand{\gentry@store}[1]{%
56   \edef\head{\noexpand\begin{group}\noexpand\gentryheader%
57     {\arabic{gentryctr}}{\gentryidx}{\gentrycode}{\gentrytitle}%
58     \noexpand\label{gentry:\gentrycode}%
59   }%
60   \global\expandafter\csname paratok\arabic{gentryctr}\endcsname=
```

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

```
61   \expandafter{\head #1%
62     \outputfootnotes@endgentry%
63     \gentryfooter\endgroup}%
64 }
```

`\gentry@footnotespergentry` 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, *ie* 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.

L<sup>A</sup>T<sub>E</sub>X 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}}\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`.

```
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.



```

74 \ifnum\outputpenalty<-\@M\else%
75   \edef\tmp@{footnotetoks\botmark}
76   \expandafter\ifcsname footnotetoks\botmark\endcsname%
77   \global\setbox255=\vbox to \vsize{%
78     \unvbox255\fill\outputfootnotes@endpage}%
79   \fi%
80 \fi%
81 \the\gentry@oldoutput
82 }\fi%

```

## 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 disparate numbers, not an especially random one.

1. Work out how many entries there are ( $N$ ). Provided `\thegentries` is called at the end, this is just the value of `gentryctr`
2. Declare a set of token registers named `\paratoks $n$` , where  $n$  is each integer  $1 \dots N$  inclusive. These will hold the contents of the entry.
3. Declare a set of macros named `\paraidx $n$` , where  $n$  is each integer  $1 \dots N$  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 `\gentrytoks $R$`  and `\gentrytoks $i$`
  - (c) If  $R \neq i$  then swap macros `\paraIdx $R$`  and `\gentryidx $i$`
5. For  $i = 1 : n$ , output token reg  $i$

Define macros `\csname paraIdx $n$ \endcsname` containing the arabic original page number to be put out on the  $n$ th output “page”

```

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

```

we can reuse `gentryctr`; we've finished this set of paras and kept the total count.

```

84 \setcounter{gentryctr}{0}%
If there are fewer than 3 entries, don't try and shuffle.
85 \ifthenelse{\gentrycount<3}{\SHUFFLE START
86 \whiledo{\not{\value{gentryctr}>\gentrycount}}{\%
87 \edef\gentryidxu{\arabic{gentryctr}}%
88 \expandafter\xdef\csname paraIdx\gentryidxu\endcsname{\gentryidxu}%
89 \stepcounter{gentryctr}%
90 }%

```

**gentryidxu** The `\gentryidxu` macro can be used inside a `gentry` to obtain the current arabic shuffled index of the entry.

**gentryidxs** The `\gentryidxs` macro can be used inside a `gentry` to obtain the arabic original unshuffled index of the entry (the first `gentry` is 1, and this counter resets after each expansion of `\thegentries`).

```

91 \if@gamebook@shuffle
92 \setcounter{rand}{\gentrycount}%
93 \addtocounter{rand}{-1}\edef%
94 \stoppoint{\arabic{rand}}%

```

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

```

95 \setcounter{gentryctr}{2}%
96 \chgrand[last=\stoppoint]%
97 \whiledo{\value{gentryctr}<\stoppoint}{%

```

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

```

98 \edef\gentryidxu{\arabic{gentryctr}}%
99 \expandafter\ifcsname fixedat\gentryidxu\endcsname%
100 \gamebook@info{Not shuffling \gentryidxu; fixed pos}%
101 \stepcounter{gentryctr}%
102 \else%
103 \gamebook@info{Shuffling \gentryidxu}%
104 \stepcounter{gentryctr}%
105 \edef\nextidx{\arabic{gentryctr}}%

```

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 `fixedat\arabic{rand}` instead.

```

106 \chgrand[first=\nextidx]%
107 \rand%
108 \expandafter\ifcsname fixedat\arabic{rand}\endcsname\else%
109 \gamebook@info{Shuffling \gentryidxu to \arabic{rand}}%
110 \macroswap{paraIdx\gentryidxu}{paraIdx\arabic{rand}}%
111 \fi%
112 \fi%
113 }%

```

Now move fixed entries into their final place:

```

114 \setcounter{gentryctr}{2}%
115 \whiledo{\not{\value{gentryctr}>\stoppoint}}{%
116   \edef\gentryidxu{\arabic{gentryctr}}%
117   \expandafter\ifcsname fixedat\gentryidxu\endcsname%
118   \expandafter\edef\expandafter\mydest\expandafter%
119     {\expandafter\csname fixedat\gentryidxu\endcsname}%
120   \gamebook@info{MOVING FIXED GAMEBOOK ENTRY INTO PLACE: \gentryidxu -> \mydest}%
121   \macroswap{paraIdx\gentryidxu}%
122   {paraIdx\expandafter\csname fixedat\gentryidxu\endcsname}%

```

Edge case: It's possible that we also have `fixedat\mydest`; in which case that would be messed up with the reshuffling. So we need to rename that to `fixedat\gentryidxu` as well, then reprocess this index

```

123   \expandafter\ifcsname fixedat\mydest\endcsname%
124   \macroswap{fixedat\gentryidxu}{fixedat\mydest}%
125   \expandafter\global\expandafter\let\csname fixedat\mydest\endcsname\@undefined%
126   \addtocounter{gentryctr}{-1}%
127   \fi%
128   \fi%
129   \stepcounter{gentryctr}%
130 }}\fi%SHUFFLE END
131 \gamebook@info{Shuffled! Gentry order:}%
132 \setcounter{gentryctr}{1}%
133 \whiledo{\not{\value{gentryctr}>\gentrycount}}{%
134   \edef\gentryidxu{\arabic{gentryctr}}%
135   \gamebook@info{\gentryidxu -> \csname paraIdx\gentryidxu\endcsname}%
136   \stepcounter{gentryctr}%
137 }%

```

We can reuse the same counter again to output

```

138 \setcounter{gentryctr}{0}%
139 \gamebook@info{Outputting \gentrycount\ gamebook entries}%
140 \whiledo{\value{gentryctr}<\gentrycount}{%

```

Use `refstepcounter` in the loop to allow `\label` to work as expected.

```

141   \refstepcounter{gentryctr}%

```

Keep last entry on its own page

```

142   \ifthenelse{\value{gentryctr}=\gentrycount}{%
143     \gamebook@beforelast%
144   }{%
145     \edef\gentryidxu{\arabic{gentryctr}}%
146     \xdef\gentryidxs{\csname paraIdx\gentryidxu\endcsname}%
147     \gamebook@info{Output gentry \gentryidxu\ of \gentrycount,%
148       original idx \gentryidxs}%

```

Output the stored entry body, stripping any extraneous space:

```

149   \the\csname paratok\gentryidxs\endcsname%
150 }%

```

Finally, we clear the registers and reset the counter in case we want to start again

```

151 \gamebook@info{All gamebook entries added to main vertical list}%
152 \setcounter{gentryctr}{1}%
153 \whiledo{\not{\value{gentryctr}>\gentrycount}}{%
154   \edef\gentryidxu{\arabic{gentryctr}}%
155   \expandafter\global\expandafter\let%
156     \csname paratok\gentryidxu\endcsname\@undefined%
157   \expandafter\ifcsname fixedat\gentryidxu\endcsname%
158     \expandafter\global\expandafter\let%
159       \csname fixedat\gentryidxu\endcsname\@undefined%
160   \fi%
161   \stepcounter{gentryctr}%
162 }%
163 \setcounter{gentryctr}{0}%
164 \eject
165 \endgroup%
166 }%

```

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

`\gentryheader`     `{\langle counterIdx\rangle}{\langle fixedIdx\rangle}{\langle code\rangle}{\langle title\rangle}`

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}`

```

167 \newcommand{\gentryheader}[4]{%
168   \noindent\textbf{\Huge\arabic{gentryctr}\large\ #4}%
169   \nopagebreak%
170   \vspace{0.3em}%
171   \nopagebreak%
172   \par%
173   \marginpar{\#3}%
174 }%

```

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

```

175 \newcommand{\gentryfooter}{%
176   \par\vspace{2em}\centerline{---}\vspace{2em plus 1in}\par%
177 }%

```

`\ignorespacesandallpars@` 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.

```

178 \def\ignorespacesandallpars@{%
179   \@ifnextchar\par%
180     {\expandafter\ignorespacesandallpars\@gobble}%
181     }%
182 }%

```

`\@footnotetext` `{\token register}` 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.  
L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X'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.

```

183 \g@addto@macro{\@makefnmark}{%
184   \ifcsname in@out\endcsname\else%
185     \mark{\arabic{\@mpfn}}%
186   \fi%
187 }%

```

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.

```

188 \g@addto@macro{\gentry@footnotespergentry}{%
189 \newcommand{\@footnotetext@save}[3]{%
190   \global\newtoks#2}%
191   \global#2={\noindent\@footnotemark{#3}}%
192 }%

```

`\@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.

```

193 \g@addto@macro{\gentry@footnotespergentry}{%
194 \newcommand{\@footnotetext@save@}[2]{%
195   \edef\@tmp@{\expandafter\arabic{\@mpfn}}%
196   \expandafter\@footnotetext@save\expandafter{\@tmp@}{#1}{#2}%
197 }%

```

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`).

```

198 \g@addto@macro{\gentry@footnotespergentry}{%

```

```

199 \renewcommand{\@footnotetext}[1]{%
200   \begingroup%
201   \def\in@out{}% flag that we're building footnotes
202   \edef\tmp{\expandafter\csname footnotetoks\arabic{\@mpfn}\endcsname}%
203   \expandafter\@footnotetext@save\expandafter{\tmp}{%
204     % This fixes up the use of \cs{footnotemark} within footnotes:
205     #1}%
206   \endgroup%
207 }}%

\outputfootnotes    {\langle maxIdx \rangle}
    Command to output the footnotes, which are just in \footnotetoks $N$  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 \@mpfn, which in the default LATEX counter, will be
    used.

208 \newcounter{fncounter}%
209 \newcommand{\outputfootnotes}[1]{%
210   \begingroup%
211   \def\in@out{}% flag that footnotes are outputting; suppresses marks
212   \setcounter{fncounter}{1}

    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.

213   \def\footnote@rule{%
    The next line comes from the TUGboat suggestions, and protect against various
    user changes

214     \leftskip=0pt\rightskip=0pt\interlinepenalty=1000%
215     \penalty-1000%
216     \vspace{1pt plus 2pt minus 0.5pt}%
217     \hspace{-0.5in}\rule{1.5in}{0.4pt}\%%
218   }%

    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

219     \outputpenalty=-\@MM\break%
220     \vbox{%
221       \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.

Next, we make sure that the token register has contents.

```
222      \expandafter\ifcsname footnotetoks\arabic{fncounter}\endcsname%
223      \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/](https://tex.stackexchange.com/questions/263733/whats-the-best-practice-way-to-test-whether-parameter-is-empty)

whats-the-best-practice-way-to-test-whether-parameter-is-empty

```
224      \expandafter\if\expandafter\relax\expandafter%
225      \detokenize\expandafter{\the\tmp@@}\relax\else%
226      \footnote@rule\gdef\footnote@rule{\ifvmode\else\par\fi}%
227      \interlinepenalty\interfootnotelinepenalty%
```

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).

```
228      \expandafter\the\tmp@@%
229      \global\expandafter\tmp@@={}%
230      \expandafter\let\tmp@@\undefined%
231      \fi\fi%
232      \stepcounter{fncounter}%
233      }%
234  }%
235  \endgroup%
236 }%
```

L<sup>A</sup>T<sub>E</sub>X 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.

(see <https://www.tug.org/TUGboat/Articles/tb11-4/tb30salomon.pdf>, page 598, to read around footnotes).

`\outputfootnotes@endgentry`

This macro is called at the end of each `gentry`. It switches into vertical mode (as ending `gentry` doesn't throw a `\break` or line end by itself), then we output all footnotes so far (by reading the footnote counter `\@mpfn`, as this is "sequential").

```
237 \newcommand{\outputfootnotes@endgentry}{%
238   \if@gamebook@footnotes%
239   \nopagebreak\ifhmode\\fi% get into vertical mode
240   \nopagebreak\outputfootnotes{\arabic{\@mpfn}}}%
241   \fi%
242 }%
```

`\noentryfoot` 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  $\text{\LaTeX}$  expands the footnote at the end of the entry, then decides to put the page split between the footnote mark and the text.

```

243 \newcommand{\noentryfoot}{\def\outputfootnotes@endgentry{}}
\outputfootnotes@endpage This is called by the output routine at the end of each page. If \botmark has a
value, then we can output all footnotes up to that index.
244 \g@addto@macro{\gentry@footnotespergentry}{
245 \newcommand{\outputfootnotes@endpage}{%
246 \expandafter\if\expandafter\relax\expandafter%
247 \detokenize\expandafter{\botmark}\relax\else%
248 \outputfootnotes{\botmark}%
249 \fi%
250 }%

```

## Change History

v1.0		v1.1
General: Initial Release . . . . .	1	General: Bugfix: edge case with
endpage: Endpage option . . . . .	3	clashing fixed-index entries . . .
footnote: Footnote option . . . . .	3	1
noshuffle: Noshuffle option . . . . .	3	gentryidxs: Access original index
verbose: Verbose option . . . . .	3	10
		gentryidxu: Access shuffled index
		10



## 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.

<b>Symbols</b>	<code>\gentry</code> . . . . . 32	<code>\if@gamebook@shuffle</code>
<code>\@@gentry</code> . . . . . 36, 43	<code>gentry</code> (environment) <u>32</u>	. . . . . 16, 18, 91
<code>\@footnotemark</code> . . . . 191	<code>\gentry@footnotespergentry</code>	<code>\ignorespacesandallpars</code>
<code>\@footnotetext</code> . . <i>13</i> , 199	. . . . . 8, 65, 68,	. . . . . 37, 53, 180
<code>\@footnotetext@save</code>	188, 193, 198, 244	<code>\ignorespacesandallpars@</code>
. . . . . 189, 196	<code>\gentry@oldoutput</code> .	. . . . . <i>13</i> , 178
<code>\@footnotetext@save@</code>	. . . . . 71, 72, 81	<code>\in@out</code> . . . . . 201, 211
. . . . . 194, 203	<code>\gentry@store</code> . . . 52, 55	
<code>\@gentry</code> . . . . . 32	<code>\gentrycode</code> . . . . .	<b>M</b>
<code>\@makefnmark</code> . . . . . 183	. . . 35, 39, <u>43</u> , 57, 58	<code>\macroswap</code> 110, 121, 124
<code>\@mpfn</code> 185, 195, 202, 240	<code>\gentrycount</code> . . . . .	<code>\mydest</code> 118, 120, 123–125
	. . . 69, 83, 85,	
	86, 92, 133, 139,	<b>N</b>
<code>\_</code> . . . . . 139, 147, 168	140, 142, 147, 153	<code>\nextidx</code> . . . . . 105, 106
<b>C</b>	<code>\gentryfooter</code> <i>12</i> , 63, 175	<code>\noentryfoot</code> . . . <i>16</i> , 243
<code>\chgrand</code> . . . . . 96, 106	<code>\gentryheader</code> . . . . .	<code>\noshuffle</code> . . . . . <u>1</u> , <u>16</u>
	. . . . . 2, <i>12</i> , 56, 167	
<b>E</b>	<code>\gentryidx</code> . . . . . 34,	<b>O</b>
<code>\eject</code> . . . . . 9, 164	38, <u>43</u> , 46, 48, 57	<code>\output</code> . . . . . 72, 73
<code>\endpage</code> . . . . . <u>1</u> , <u>7</u>	<code>\gentryidxs</code> . . . . .	<code>\outputfootnotes</code> . .
environments:	41, <u>91</u> , 146, 148, 149	. . . <i>14</i> , 209, 240, 248
<code>gentry</code> . . . . . <u>32</u>	<code>\gentryidxu</code> . . 40, 87,	<code>\outputfootnotes@endgentry</code>
<b>F</b>	88, <u>91</u> , 98–100,	. . . <i>15</i> , 62, 237, 243
<code>\footnote</code> . . . . . <u>1</u> , <u>11</u>	103, 109, 110,	<code>\outputfootnotes@endpage</code>
<code>\footnote@rule</code> 213, 226	116, 117, 119–	. . . . . <i>16</i> , 78, 245
	122, 124, 134,	
<b>G</b>	135, 145–147,	<b>S</b>
<code>\gamebook@beforelast</code>	154, 156, 157, 159	<code>\stoppoint</code> 94, 96, 97, 115
. . . . . 7, 9, 143	<code>\gentrytitle</code> . . . . 44, 57	
<code>\gamebook@info</code> 3, 5, 6,	<b>I</b>	<b>T</b>
14, 19, 100, 103,	<code>\if@gamebook@footnotes</code>	<code>\thegentries</code> . . . . . 8, 66
109, 120, 131,	11, 13, 68, 73, 238	
135, 139, 147, 151	<b>V</b>	<code>\verbose</code> . . . . . <u>1</u> , <u>3</u>