Welcome to the **RPGTEX** package. This LaTeX package is designed to allow users to flexibly typeset documents associated with Role Playing Games such as *Dungeons & Dragons* – and many more besides. This packages defines a central engine: `rpgcore` which define a number of useful functions and classes, and a flexible set of `themes` which control how those commands are rendered in the final document.

Attribution & License

This package would not have been possible without the team who developed its predecessor, the 'DND 5e LateX Template' . That code was released under an MIT license, the text of which can be found in the LICENSE file. `rpgtex` is released under the same license.

Contents

PART I
rpgtex Core

Chapter 1: Installation & Usage

Getting `rpgtex`

There are a number of different ways to acquire `rpgtex`. Once you have installed it, it is vital to ensure that it is properly configured (see below).

texmf Installation

The simplest way to use `rpgtex` is to install it on the `texmf` path, where the compiler can automatically find it:

```
git clone
    https://github.com/DrFraserGovil/rpgtex.git
    "$(kpsewhich -var-value
    TEXMFHOME)/tex/latex/rpgtex"
```

This will clone the repository into yourLaTeX path.

Indirect Installation

If you want to tinker with `rpgtex` – such as by creating a new theme – it is helpful to have it in a more accessible location. Clone the repository into a location of your choice:

```
git clone
    https://github.com/DrFraserGovil/rpgtex.git
    ~/your/rpgtex/directory
```

You then have two options to make the package visible to the compiler:

Use TEXINPUTS

Setting the environment variable `TEXINPUTS` allows the compiler access:

```
TEXINPUTS=~/your/rpgtex/directory/::
```

(Or similar commands, depending on your shell – in `fish` you would call `set TEXINPUTS dir`).

Use Symlinks

You can symlink the install location to the texmf directory, allowing the compiler to act as if you had performed the texmf installation:

```
ln -sf ~/your/rpgtex/directory "$(kpsewhich
    -var-value TEXMFHOME)/tex/latex/rpgtex"
```

Overleaf (Not recommended!)

We do not recommend using Overleaf since the free-tier subscription has reduced compilation times drastically, making compiling documents using complex packages such as this one extremely difficult. Nevertheless:

1. Download this GitHub repository as a ZIP archive using the Clone or download link above.
2. On Overleaf, click the New Project button and select Upload Project. Upload the ZIP archive you downloaded from this repository.
3. Manually create the file `rpg-config.cfg` with the contents "`\edef\RpgPackagePath{../}`". This replaces the configuration step described below.

Configuring `rpgtex`

Wherever one installs `rpgtex` from, it is vital that it is properly configured. From within the `rpgtex`-root directory, call:

```
./configure
```

Or – if one is (reasonably!) wary about running arbitrary executables – manually create the relevant file:

```
cd <rpgtex root directory>
cmd="\edef\RpgPackagePath{$(pwd)}"
echo $cmd >> core/rpg-config.cfg
```

Why is configuration necessary?

TeX is generally set up so that when a file calls `include` or `input` it is possible to use filepaths relative to the package itself. `rpg.sty` can call `\input{core/font.sty}` and it will know to first check for the file relative to rpg.sty; even if the package resides within the texmf path and the user has no idea where `rpgroot/rpg.sty`, or `rpgroot/core/font.sty`, are.

An annoying exception to this is fonts and typefaces. `xelatex` searches for fonts based on *filepaths relative to the current working directory* – or from those installed in as system fonts.

Since `rpgtex` includes several (license free) typefaces as part of the provided themes, this poses a problem. We must either require that:

1. `rpgtex` documents can only be prepared in restricted locations relative to the install location of `rpgtex`.
2. Users must identify and specify the `rpgtex` root path when preparing a document
3. Users must install the provided fonts to the system path
4. `rpgtex` must be configured to know 'where it is', and so provide an absolute filepath to the internal fonts.

The Configuration step is the most portable and easiest-to-use of these options.

Without a `core/rpg-config.cfg` file, any document which includes `rpgtex` will fail to compile.

Package & Class Usage

`rpgtex` can be used either as a standalone package, or as part of a number of classes

Standalone Package

The standalone package can be used directly by including the `rpgtex` package:

```
\documentclass{arbitrary-class}

\usepackage[options]{rpgtex}

\begin{document}
....
```

This will load only the core commands into the document, and (unless called explicitly) no themes will be imported. Using the package in this way does not activate any of the commands which change the overall geometry, background or headers of the document.

Classes

`rpgtex` can also be loaded through a number of classes which drastically alter the appearance of the document, defining new geometries backgrounds and adding headers.

The provided classes are:

1. `rpgbook` (page 10). Based on the standard book class, this is designed for larger RPG documents.
2. `rpghandout` (page 11). Based on the article class, this is designed for shorter documents
3. `rpgcard` (page 12). A small-document class designed for creating modular 'handout' cards for items, spells or abilities.

Compiling

`rpgtex` uses the `fontspec` package to allow custom fonts, and therefore requires compiling with `xelatex` or `luatex`:

```
xelatex main.tex  #works
luatex main.tex   #works
pdflatex main.tex #fails
```

Chapter 2: Core Commands

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Redefinitions of Common Commands

`\part`
`\part *`
`{o m}`

Defines a wrapper around the standard `part` command that allows for tikz-based custom page formatting

Syntax

```
\part (*)[<image>]{<part-name>}
```

Behaviour

There are three distinct behaviours that can be exhibited, depending on the presence or absence of the `*`, and the presence and value of `<image>`.

Command	Behaviour

`\part *{partname}`
`\part *[<any text>]{partname}`
`\part [none]{partname}`
	Uses original `part` command defined by underlying class.

`\part {partname}`	Calls `RpgDrawPartPage` on a blank background.
`\part [path/to/image]{partname}`	Places the corresponding image as a full-page background, and then calls `RpgDrawPartPage`.

`RpgDrawPartPage` (page 7) is a Theme Function, which executes a series of tikz functions to place the part title according to the theme specifications.

Document Commands

`\RpgDice {m}` Evaluates expressions of the form $ndx + m$, and outputs using a theme-dependent layout.

Syntax

```
\RpgDice {<dice-expression>}
```

Behaviour

Uses regular expressions to extract and evaluate the dice expression. The `dice-expression` must follow the following format:

1. It must contain either 'd' or 'D' (the 'dice symbol')
2. The dice symbol must be immediately followed by a single number (the 'dice size')
3. The dice symbol may optionally be prefixed by a single number (the 'dice count')
4. The first (non-whitespace) character must be either the dice count (if present) or the dice symbol
5. The dice size must be followed by either a '+', '-', or the end of the expression.
6. After this, any number of standard numeric expressions may follow. This expression will be evaluated into a single 'modifier'.

Example	Output
`\RpgDice {1d6-2}`	1d6-2
`\RpgDice {2D6 + 3*2^2}`	2d6+12
`\RpgDice {1d16}`	1d16
`\RpgDice {d8-3}`	d8-3

Themes

`RpgDice` is neat, but not necessarily impressive by itself. The true power of the expression is that it calls `RpgDiceFormat` (page 6) on the internal values. This is a value which can be overwritten by themes, and allows (for instance), the `dnd` theme to compute mean values, as is found in D&D monster stat blocks:

Example (with `\RpgSetTheme {dnd}`)	Output
`\RpgDice {1d6-2}`	1 (1d6-2)
`\RpgDice {2D6 + 3*2^2}`	19 (2d6+12)
`\RpgDice {1d12}`	6 (1d12)
`\RpgDice {d83-3}`	39 (d83-3)

`\RpgOrdinal {o m}` Converts a numeric value to the corresponding ordinal.

Syntax

```
\RpgOrdinal [<command>]{<count>}
```

Behaviour

The command outputs the `count` followed by the english abbreviations for the corresponding ordinal. The optional `command` argument is inserted between the numeral and the suffix, allowing for the customisation of appearances.

Example	Output
`\RpgOrdinal {1}`	1st
`\RpgOrdinal {2}`	2nd
`\RpgOrdinal {13}`	13th
`\RpgOrdinal [\textsuperscript ]{7}`	7^{th}
`\RpgOrdinal [\textbf ]{133}`	133**rd**
`\RpgOrdinal [<arbitrary text>]{133}`	133<arbitrary text>rd

Note that due to a lack of brace-capturing, it is not possible to chain multiple commands..

`\RpgPage {O{t} m}`

Outputs the current page reference for a label, with an option to enclose it in specific brackets or parentheses.

Syntax

```
\RpgPage [t/p/b/c]{<label-reference>}
```

Behaviour

The optional arguments wrapping of the main reference. The options are:

t (default) No wrapping

p (parentheses)

b [square brackets]

c {curly braces}

An invalid input resolves to `?page~\pageref {<ref}?`.

Example	Output
`\RpgPage {example:current page}`	page 6
`\RpgPage [p]{example:current page}`	(page 6)
`\RpgPage [b]{example:current page}`	[page 6]
`\RpgPage [c]{example:current page}`	{page 6}
`\RpgPage [(error)]{example:current page}`	?page 6?

`\RpgPlural {o m m}`

Generates grammatically correct plural forms of a word based on a given count.

Syntax

```
\RpgPlural [<custom-plural>]{count}{<text>}
```

Behaviour

The command outputs the count followed by the value of `<text>`. For a count of 1, the command then finishes. For any other count, it appends an "s", pluralizing the text.
The optional argument `[<custom-plural>]` overrides the default logic, allowing for irregular plurals.

Example	Output
`\RpgPlural {1}{hat}`	1 hat
`\RpgPlural {2}{hat}`	2 hats
`\RpgPlural [octopodes]{1}{octopus}`	1 octopus
`\RpgPlural [octopodes]{359}{octopus}`	359 octopodes

Theme Commands

`\RpgDiceFormat {m m m}`

Prints the values computed by `RpgDice` (page 5)

Syntax

```
\RpgDiceFormat {<dice-count>}{<dice-size>}{<added bonus>}
```

Behaviour

This function is used by theme designers to determine how `RpgDice` is rendered. The default option is: `\RpgDiceFormat {m m m} { #1d#2 #3}`, such that `RpgDice{ndx + a + b}` gives "ndx + c", where c is the numerical value of a+b, with an additional check to see if `#3` is equal to 0, in which case it is not printed (so as not to '1d6 + 0').
The dnd implementation performs a more advanced operation, computing the average value of the roll, and formatting that first, to replicate the format used by monster stat blocks.
`RpgDiceFormat` is loaded in both `layout` and non-`layout` calls.

`\RpgDrawPartPage {m}`

Uses Tikz to draw a custom part page when activated by `\part` (page 4).

Syntax

```
\RpgDrawPartPage {<part title>}
```

Behaviour

This function is mostly used to determine where to place the part name on the page, and what embellishments accompany it. The command is called from within an existing tikz environment with the `remember,overlay` options active, allowing for page coordinates (i.e. current page.north) to be used.
The default `part` command allows a user to specify a background image for their part page – it is not necessary to provide one within the drawing command.

`\RpgLayoutOnly {m}`

Executes the contents of the command if `layout` mode is active.

Syntax

```
\RpgLayoutOnly {<content-to-execute>}
```

Behaviour

If the internal value `\l__rpg_layout_bool` is True, then `content-to-execute` is run, otherwise it is ignored.
This command is primarily used by theme developers and document class files to conditionally load or activate modules based on whether the package was loaded via a document class (layout mode active) or directly via `\usepackage {rpgtex}`.

`\RpgSetTheme {m}`

Activates a chosen theme.

Syntax

```
\RpgSetTheme {<theme-name>}
```

Behaviour

Searches for the file `<theme-path>/<theme-name>/<theme-name>.cfg`, and inputs it. If this is a properly configured theme file, then it activates the chosen theme given the current global parameters. If the file does not exist, throws an error.
If `\l__rpg_layout_bool` is True, the command automatically inserts `\clearpage`, as required to ensure the old headers are not overwritten by the new theme.
`<theme-path>` is modified via `RpgSetThemePath`.

`\RpgSetThemeColor {m}`

Sets the `themecolor`, and simultaneously updates the co-varying colors (page 8).

Syntax

```
\RpgSetThemeColor {color-name}
```

Behaviour

If `color-name` specifies a valid color, then the value of `themecolor` is updated, as well as a number of other colors (`tipcolor`, `sidebarcolor` and `tablecolor`) which are set to be equal to the themecolor by default.
Of the rpg-provided colors, only `narrationcolor` is unaffected by this command.

`\RpgSetThemePath {m}`

Changes the value of the theme path searched for by `RpgSetTheme`

Syntax

```
\RpgSetTheme {<path-name>}
```

Behaviour

Updates an internal variable to be equal to the input value; does not check if the theme path is valid or not. Useful if you wish to create a new theme outside of the `rpgtex` file structure.

Variables

Colo(u)rs

`rpgtex` by default defines five colors[1] which are used for different elements:

themecolor A 'basic color' which is (by default) equal to the following three colors:

1. **sidebarcolor** The background color of the `RpgSidebar` environment
2. **tablecolor** The background color of every other row in an `RpgTable`
3. **tipcolor** The background color of the `RpgTip` environment

narrationcolor The background color of the `RpgTip` environment

Calling `\RpgSetThemeColor` (page 7) updates the value of `themecolor`, as well as the three 'co-varying' colors (i.e. everything except `narrationcolor`). When `printmode` is active `\RpgSetThemeColor{white}` is called, making environments transparent.

[1] Yes, I hate myself, but we're going with the code-based spelling.

PART II
rpgtex Classes

Chapter 3: rpgbook Class

Chapter 4: rpghandout Class

Chapter 5: rpgcard Class

PART III
Themes

Chapter 6: `default` Theme

Chapter 7: DND Theme

Chapter 8: scifi Theme