

The `rpgicons` package

A set of high-quality icons for use in notes for tabletop role-playing games

Jasper Habicht *

Version 2.9.2, released on 23 November 2026

---

1 Introduction

The `rpgicons` package provides a set of high-quality icons for use in notes for tabletop role-playing games. The icons are meant to be used in the body text, but they can also be used in other contexts such as graphics or diagrams.

The package comes in two variants, a L3 variant based on the `l3draw` package which is loaded per default and a PGF variant based on PGF/Ti*k*Z.

2 Loading the package

To install the package, copy the relevant package files `rpgicons.sty`, `rpgicons-l3.sty` and `rpgicons-pgf.sty` as well as `rpgicons.lua` into the working directory or into the `texmf` directory. You may want to use a TeX package manager for this. After the package has been installed, the `rpgicons` package is loaded by calling `\usepackage{rpgicons}` in the preamble of the document.

The file `rpgicons.sty` acts as a wrapper that loads either the file `rpgicons-l3.sty` or `rpgicons-pgf.sty` depending on the variant of the package that is used. Therefore, if only one of the variants should be used, technically speaking it is sufficient to only install on of these two files and load the package using `\usepackage{rpgicons-l3}` or `\usepackage{rpgicons-pgf}` respectively. Note, however, that the file `rpgicons.lua` needs to be installed if the L3 variant is used with LuaLaTeX.

The package can be loaded with one of the following options.

`l3`

The L3 variant of the package is loaded by default. To load it explicitly, the package can be loaded using the option `l3`.

* E-mail: mail@jasperhabicht.de. I am grateful to Joseph Wright who fixes bugs at an astonishing speed, to Ulrike Fischer and David Carlisle for their help in adding tagging support to this package, to Max Chernoff for inspiring and guiding me in using Lua and to all other contributors to this package.

pgf

To load the PGF variant of the package the package needs to be loaded with the option `pgf`.

2.1 Dependencies

The L3 variant of the package loads the `l3draw` package. When compiled with LuaLaTeX, the file `rpgicons.lua` needs to be installed.

The PGF variant of the package loads the `tikz` package which in turn loads the `xcolor` package. To make use of specific options these packages provide, you need to load the packages with the relevant options beforehand or explicitly pass the relevant options to the package using a command such as `\PassOptionsToPackage{svgnames}{xcolor}`.

2.2 Generating SVG icons

To generate an SVG from the LaTeX source, use `\documentclass[dvisvgm]{standalone}` and compile with `latex` and then run `dvisvgm -b papersize` on the compiled DVI file.

The below code example can be compiled with `latex` and `dvisvgm -b papersize -p 1-` applied to the resulting DVI file to generate one SVG file for each graphic inside the `icon` environment:

```
\documentclass[dvisvgm, multi=icon]{standalone}
\usepackage{rpgicons}

\begin{document}
    \begin{icon}%
        \ability{charisma}%
    \end{icon}
    \begin{icon}%
        \class{barbarian}%
    \end{icon}
\end{document}
```

3 Main user commands

Regardless of the variant of the package, a set of user commands is always available. These are described in the following. Depending on the use of the L3 or the PGF variant, certain specific commands or options are available that are explained in the following sections in further detail.

Because of the way the package defines the icons, each of the user commands below described can actually be used together with every shape. However, the combinations of shapes and commands as described in the following subsections are preferable.

Note that shape names ending with `alt` have been deprecated. Instead, use the `variant` key to select variants of icon shapes as described below.

3.1 Command `\die`

\die[‹style›]{‹shape›}[‹options›]{‹integer›}

The command `\die` typesets an icon to depict a die with a certain number of sides.[1] Additionally,

1 The code for the dice shapes is based on the idea of John Kormylo (see: https://tex.stackexchange.com/a/144474/47927).

icons exist for a two-sided die (which would be equivalent to a coin) and for a hundred-sided die (which typically comes in the shape of a sphere). There is also a special icon for a fudge die.

For the six-sided die, nine additional shapes exist representing the values one to nine as pips. Also, additional shapes exist representing the plus or minus side of a fudge die.

The command takes two mandatory arguments, the first of which describes the shape (see the lists below) and the second taking an integer that is placed in front of the shape. For example, `\die{eightside}{2}` results in 2  (meaning that two eight-sided dice are rolled). If tagging is enabled, the default replacement text of this command is "2 eightside".

The command also takes two optional arguments, the second of which can take additional options to style the icon or to select different variants of the icon shape. The options affect the shape, but not the integer when it is typeset before the icon. The usable options differ depending on the package variant. See the relevant sections below.

The first optional argument can take the value `normal` or `large`, `normal` being the default value. With `large` given as argument, the icon is drawn larger and the integer is typeset inside the shape. As an example, `\die[large]{eightside}{2}` results in . Note that the integer will always be placed on top of the shape, even if the shape does not have an open center which is the case with the `fudge` shapes or the shapes featuring pips.

Command	Icon	Shape	Variant
`\die`		`twoside` or `d2`	
		`fourside` or `d4`	
		`sixside` or `d6`	
		`eightside` or `d8`	
		`tenside` or `d10`	
		`twelveside` or `d12`	
		`twentyside` or `d20`	
		`hundredside` or `d100`	
		`fudge`	
		`sixside one`	
		`sixside two`	
		`sixside three`	
		`sixside four`	
		`sixside five`	
		`sixside six`	
		`sixside seven`	
		`sixside eight`	
		`sixside nine`	
		`fudge plus`	
		`fudge minus`	

3.2 Commands `\ability` and `\saving`

\ability[‹style›]{‹shape›}[‹options›]

The command `\ability` typesets an icon depicting an ability of a character. The abilities are represented by animal-like shapes. The relevant shape should be given as mandatory argument to the command. The second optional argument can take additional options to style the icon.

The first optional argument can take the value `positive` or `negative`, `positive` being the default value. With `negative` given as argument, the icon is drawn negative inside a circle. As an example, `\ability[negative]{charisma}` results in .

\saving[‹style›]{‹shape›}[‹options›]

The command `\saving` typesets an icon with the relevant `\ability` icon inside a small shield. It can take the same values for the mandatory argument as the `\ability` command. The optional argument can take additional options to style the icon.

The first optional argument can take the value `normal` or `empty`, `normal` being the default value. With `empty` given as argument, the icon inside the shield is not typeset. In this case, the mandatory argument can be left empty. As an example, `\saving[empty]{}` results in .

Command	Icon	Shape	Variant
`\ability`		`strength`	
		`dexterity`	
		`dexterity`	1
		`constitution`	
		`intelligence`	
		`wisdom`	
		`wisdom`	1
		`charisma`	
		`resilience`	
		`sanity`	
		`perception`	
		`luck`	
		`armour` or `armor`	
		`proficiency`	
`\saving`		`strength`	
		`dexterity`	
		`dexterity`	1
		`constitution`	
		`intelligence`	
		`wisdom`	
		`wisdom`	1
		`charisma`	
		`resilience`	
		`sanity`	
		`perception`	
		`luck`	
		`armour` or `armor`	
		`proficiency`	

3.3 Command `\spell`

```
\spell{‹shape›}[‹options›]
```

The command `\spell` typesets an icon depicting the effect of a spell or how it is to be effected. The optional argument can take additional options to style the icon.

Command	Icon	Shape	Variant
`\spell`		`linear`	
		`conic`	
		`quadratic`	
		`cubic`	
		`spheric`	
		`cylindric`	
		`emanation`	
		`verbal`	
		`somatic`	
		`material`	
		`ritual`	
		`focus`	

3.4 Command `\spellschool`

```
\spellschool[‹style›]{‹shape›}[‹options›]
```

The command `\spellschool` typesets an icon that represents the school a spell belongs to. The second optional argument can take additional options to style the icon.

The first optional argument can take the value `negative` or `positive`, `negative` being the default value. Per default the icon is drawn in white inside a filled escutcheon. With `positive` given as argument, the icon as well as the escutcheon are drawn in the currently selected color. As an example, `\spellschool[positive]{evocation}` results in .

Command	Icon	Shape	Variant
`\spellschool`		`abjuration`	
		`conjuration`	
		`divination`	
		`enchantment`	
		`evocation`	
		`illusion`	
		`necromancy`	
		`transmutation`	

3.5 Commands `\damage`, `\attack` and `\condition`

```
\damage{‹shape›}[‹options›]
```

The command `\damage` typesets an icon depicting the damage of an attack. The icon is printed inside a circle. The optional argument can take additional options to style the icon.

```
\attack{‹shape›}[‹options›]
```

The command `\attack` typesets an icon depicting the kind of an attack. The optional argument can take additional options to style the icon.

\condition{‹shape›}[‹options›]

The command `\condition` typesets an icon depicting a condition of a character. The optional argument can take additional options to style the icon.

Command	Icon	Shape	Variant
`\damage`		`acid`	
		`bludgeoning`	
		`cold`	
		`fire`	
		`force`	
		`lightning`	
		`necrotic`	
		`necrotic`	1
		`piercing`	
		`poison`	
		`psychic`	
		`radiant`	
		`slashing`	
		`thunder`	
		`healing`	
`\attack`		`melee`	
		`melee`	1
		`ranged`	
		`magic`	
		`singlehanded`	
		`doublehanded`	
`\condition`		`buff`	
		`blinded`	
		`charmed`	
		`deafened`	
		`exhausted`	
		`frightened`	
		`grappled`	
		`incapacitated`	
		`invisible`	
		`paralyzed`	
		`petrified`	
		`poisoned`	
		`prone`	
		`restrained`	
		`restrained`	1
		`stunned`	
		`unconscious`	
		`hearing`	
		`seeing`	

3.6 Commands `\class` and `\alignment`

```
\class[‹style›]{‹shape›}[‹options›]
```

The command `\class` typesets an icon depicting a class of a character. The optional argument of the command can take additional options to style the icon.

The first optional argument can take the value `negative` or `positive`, `positive` being the default value. With `positive` given as argument, the icon is drawn in white inside a filled frame. As an example, `\class[negative]{barbarian}` results in .

```
\alignment[‹style›]{‹shape›}[‹options›]
```

The command `\alignment` typesets an icon depicting an alignment of a character. The optional argument of the command can take additional options to style the icon.

The first optional argument can take the value `negative` or `positive`, `positive` being the default value. With `positive` given as argument, the icon is drawn in white inside a filled frame. As an example, `\alignment[negative]{lawful good}` results in .

Command	Icon	Shape	Variant
`\class`		`artificer`	
		`artificer`	1
		`barbarian`	
		`bard`	
		`bard`	1
		`cleric`	
		`cleric`	1
		`cleric`	2
		`druid`	
		`druid`	1
		`druid`	2
		`fighter`	
		`gunslinger`	
		`monk`	
		`monk`	1
		`paladin`	
		`paladin`	1
		`psion`	
		`psion`	1
		`ranger`	
		`ranger`	1
		`rogue`	
		`rogue`	1
		`sorcerer`	
		`sorcerer`	1
		`warlock`	
		`wizard`	
		`wizard`	1
`\alignment`		`lawful good`	
		`neutral good`	
		`chaotic good`	
		`lawful neutral`	
		`true neutral`	

Command	Icon	Shape	Variant
`\alignment` (cont.)		`chaotic neutral`	
		`lawful evil`	
		`neutral evil`	
		`chaotic evil`	

3.7 Command `\currency`

```
\currency{‹shape›}[‹options›]{‹integer›}
```

The command `\currency` typesets an icon depicting a value of a coin. The optional argument can take additional options to style the icon.

The command provides a second mandatory argument that accepts an integer that is placed in front of the shape. For example, `\currency{gold}{7}` results in 7 . If tagging is enabled, the default replacement text of this command is "7 gold".

Command	Icon	Shape	Variant
`\currency`		`copper`	
		`silver`	
		`gold`	
		`electrum`	
		`platinum`	
		`gem`	
		`jewellery` or `jewelry`	
		`jewellery` or `jewelry`	1

3.8 Command `\movement`

```
\movement[‹style›]{‹shape›}[‹options›]{‹integer›}
```

The command `\movement` typesets an icon depicting a direction of movement.

The command takes two mandatory arguments, the first of which describes the shape (see the lists below) and the second taking an integer that is placed in front of the shape. For example, `\movement{forward}{5}` results in 5 . If tagging is enabled, the default replacement text of this command is "5 forward".

The command also takes two optional arguments, the second of which can take additional options to style the icon or to select different variants of the icon shape. The options affect the shape, but not the integer when it is typeset before the icon. The usable options differ depending on the package variant. See the relevant sections below.

The first optional argument can take the value `normal` or `large`, `normal` being the default value. With `large` given as argument, the icon is drawn larger and the integer is typeset inside the shape. As an example, `\movement[large]{forward}{5}` results in .

Command	Icon	Shape	Variant
`\movement`		`upward`	
		`downward`	
		`leftward`	

Command	Icon	Shape	Variant
`\movement` (cont.)	⇨	`rightward`	
	⇧	`forward`	
	⇩	`backward`	

4 Specifics of the L3 package variant

The L3 variant of the package which uses the `l3draw` package is loaded by default or explicitly by either calling `\usepackage[l3]{rpgicons}` or `\usepackage{rpgicons-l3}` in the preamble of the document after having installed the files `rpgicons.sty` and `rpgicons-l3.sty`. The `l3draw` package is an experimental package that provides only basic drawing functionality. The L3 variant thus only supports a certain set of options for styling the icons.

The L3 variant of the package does not load the `xcolor` package but makes use of the `l3color` module which uses a similar syntax like the `xcolor` package. Color definitions made using the `l3color` module are not directly usable via commands provided by the `xcolor` package. Therefore, setting a color using the `\color` macro provided by the `xcolor` package won't affect the color of the icons.

It is possible, however, to call `\DocumentMetadata{}` before `\documentclass` and then use `\definecolor` provided by the `xcolor` package to define a color which can then be used by the L3 variant of the package.

Transparency requires the management of certain PDF settings. Therefore, it is necessary to call `\DocumentMetadata{}` before `\documentclass` when using the L3 variant of the package.

It is also possible to call `\RequirePackage{pdfmanagement}` before `\documentclass` instead of `\DocumentMetadata{}` to enable the above described color and transparency management.

4.1 Icon commands

variant
variant={‹integer›}

The main user commands (as well as the underlying commands on which these commands are based as described below) accept as option the key `variant` which accepts an integer as value to select variants for the relevant shape. See the icon lists above for the respective variants. For example, the command `\ability{dexterity}[variant=1]` results in 🐱.

Setting the key `variant` without value is equivalent to setting its value to `1`. Selecting a non-existent variant selects the default shape.

With the following setting, all occurrences of the `necrotic` shape are set to the relevant variant.

```
\rpgiconsset{
    every necrotic={
        variant=1
    }
}
```

\RPGIconsUseIcon[‹options›][‹integer›]{‹shape›}
\RPGIconsUseIcon*[‹options›][‹integer›]{‹shape›}

`\RPGIconsUseIcon` is the primary command to typeset icons using the L3 variant of the package. The commands `\die`, `\ability`, `\saving`, `\spell`, `\spellschool`, `\damage`, `\attack`, `\condition`, `\class`, `\alignment`, `\currency` and `\direction` are based on this command. This command allows to combine all shapes with all frames and with either a positive or a negative color scheme.

For example, `\RPGIconsUseIcon[scale=.675, stroke=blue][2]{tenside}` is equivalent to and thus results in the same output as `\die[large]{tenside}[stroke=blue]{2}` and `\RPGIconsUseIcon*[frame=ability, scale=.333, fill=red]{charisma}` results in the same output as `\ability[negative]{charisma}[fill=red]`. On the other hand, it is also possible to create icons that can't be typeset using the above standard commands, such as `\RPGIconsUseIcon*[frame=saving, scale=.333, fill=blue]{barbarian}` which will result in .

The `\RPGIconsUseIcon` command has a starred version and two optional arguments as well as one mandatory argument. The mandatory argument holds the shape of the icon. The second optional argument can be used to add an integer (or any text) that is placed on top of the icon, for example when used with shapes for dice.

The starred version of the command is used to fill a frame with color instead of drawing its outline. Frames can be put around the shape via the relevant `frame` option.

The `\RPGIconsUseIcon` command supports PDF tagging. If tagging is activated by using `\DocumentMetadata{tagging=on}`, a replacement text is automatically added to the relevant icon which can be copied to the clipboard and can be read by screen readers. Note that tagging support depends on the used PDF reader.

```
\RPGIconsDie[⟨style⟩]{⟨shape⟩}[⟨options⟩]{⟨integer⟩}
\RPGIconsAbility[⟨style⟩]{⟨shape⟩}[⟨options⟩]
\RPGIconsSaving[⟨style⟩]{⟨shape⟩}[⟨options⟩]
\RPGIconsSpell{⟨shape⟩}[⟨options⟩]
\RPGIconsSpellschool[⟨style⟩]{⟨shape⟩}[⟨options⟩]
\RPGIconsDamage{⟨shape⟩}[⟨options⟩]
\RPGIconsAttack{⟨shape⟩}[⟨options⟩]
\RPGIconsCondition{⟨shape⟩}[⟨options⟩]
\RPGIconsClass[⟨style⟩]{⟨shape⟩}[⟨options⟩]
\RPGIconsAlignment[⟨style⟩]{⟨shape⟩}[⟨options⟩]
\RPGIconsCurrency{⟨shape⟩}[⟨options⟩]{⟨integer⟩}
\RPGIconsDirection{⟨shape⟩}[⟨options⟩]{⟨integer⟩}
```

This set of commands are the basic commands defined by the L3 variant of the package on which the commands `\die`, `\ability`, `\saving`, `\spell`, `\spellschool`, `\damage`, `\attack`, `\condition`, `\class`, `\alignment`, `\currency` as well as `\direction` are based. This set of commands can be used in cases where another package defines one of these user commands. These user commands are exact copies of this set of commands.

4.2 Icon options

The `\RPGIconsUseIcon` command and the commands `\die`, `\ability`, `\saving`, `\spell`, `\spellschool`, `\damage`, `\attack`, `\condition`, `\class`, `\alignment`, `\currency` as well as `\direction` can be used with certain options that each consist of a key-value pair and can be combined. These options should be used directly without wrapping them inside the `style` option, when used with the `\RPGIconsUseIcon` command or the other commands based on this command.

For example, `\die{eightside}[color=blue, line width=0.8pt]{2}` would result in 2 .

```
frame={‹string›}
```

With the `frame` option, one of six different frames can be selected that are drawn around the shape of the icon. The values `ability` and `damage` draw a circle around the shape. The value `saving` draws a rounded shield and the value `spellschool` draws an angular shield around the shape. The value `class` dtaws a square and the value `alignment` a hexagon around the shape. The commands `\ability`, `\saving`, `\spellschool`, `\damage`, `\class` and `\alignment` make use of the relevant frame. Because the regular commands already implicitly select the relevant frame (or no frame), using the `frame` option is only recommended together with the `\RPGIconsUseIcon` command.

For example, `\RPGIconsUseIcon[frame=class, scale=0.333]{charisma}` would result in .

```
color={‹color›}
stroke={‹color›}
fill={‹color›}
text={‹color›}
background={‹color›}
```

```
opacity={‹float›}
stroke opacity={‹float›}
fill opacity={‹float›}
text opacity={‹float›}
background opacity={‹float›}
```

The `color` option sets the color of strokes, fills and text in general while the `stroke` option, the `fill` option and the `text` option set the color only for strokes, fills or text respectively.

Similarly, the `opacity` macro sets the opacity generally, while the options `stroke opacity`, `fill opacity` and `text opacity` allow for setting the opacity of strokes, fill and text separately.

Some icons can be used with a negative color scheme where the icon is drawn negatively inside a filled shape. Per default, the icons are drawn in white in such cases, but it might be desirable that the icons are in the same color as the background. To this end, the `background` option sets the color of the shape when it is printed over a filled frame which can be achieved by setting the `negative` option for the `\ability`, `\spellschool`, `\class` or the `\alignment` command or using the starred version of the `\RPGIconsUseIcon` command.



```
\colorbox{blue!50}{%
    \ability[negative]{charisma}
        [scale=2]%
}

\colorbox{blue!50}{%
    \ability[negative]{charisma}
        [scale=2, background=blue!50]%
}
```

```
line width={‹dimension›}
line cap={‹string›}
line join={‹string›}
miter limit={‹float›}
```

The option `line width` sets the line width for strokes. The default line width is the default line width of Ti*k*Z or the `l3draw` package respectively which in both cases is 0.4 pt.

The option `line cap` takes as value one of `butt`, `rectangle` or `round` and sets the style of stroke terminals to butt caps, rectangle caps (projecting square caps) or round caps respectively. Per default, round caps are set.

The option `line join` takes as value one of `bevel`, `miter` or `round` and sets the style of stroke joins to bevel joins, miter joins or round joins respectively. Per default, round line joins are set.

The option `miter limit` takes as value floating point number and sets the miter factor of lines joined as a miter. The default factor is 10.

```
scale={‹float›}
scale inner={‹float›}
rotate={‹float›}
```

Using the `scale` and `rotate` options, the shape can be scaled and rotated (in degrees).

The `scale inner` option can be used to change the scaling of the icon placed inside a frame when using the `\ability`, `\saving`, `\spellschool`, `\damage`, `\class` and `\alignment` macros. The default value is 0.675.



```
\damage{petrified}
    [scale=2]

\damage{petrified}
    [scale=2, scale inner=0.5]
```

Note that the keys related to transformation, that is `scale`, `scale inner` and `rotate`, add to the current value and do not replace it. For example, `scale=0.5, scale=0.5` results in a scale factor of 0.25 and `rotate=10, rotate=10` results in a rotation of 20 degrees.

```
every die={‹options›}
every ability={‹options›}
every saving={‹options›}
every spell={‹options›}
every spellschool={‹options›}
every damage={‹options›}
every attack={‹options›}
every condition={‹options›}
every class={‹options›}
every alignment={‹options›}
every currency={‹options›}
every direction={‹options›}
every ‹shape›={‹options›}
```

Styles following the pattern `every` followed by a space and the name of the command or the shape can be used to apply styles to every instance of this command or shape.

For example, `\rpgiconsset{every die={color={red}}}` can be used to draw in red all icons created using the `\die` command.

Calling `\rpgiconsset{every charisma={color={red}}}` will draw every instance of the `charisma` shape in red.

```
every die add={‹options›}
every ability add={‹options›}
every saving add={‹options›}
every spell add={‹options›}
every spellschool add={‹options›}
every damage add={‹options›}
every attack add={‹options›}
every condition add={‹options›}
every class add={‹options›}
every alignment add={‹options›}
every currency add={‹options›}
every direction add={‹options›}
every ‹shape› add={‹options›}
```

Styles following the pattern `every` followed by a space and the name of the command or the shape followed by another space and `add` work in the same way as the styles described above, but instead of overwriting existing styles, they append the relevant styles.

4.3 Setting options globally

```
\rpgiconsset
```

Apart from setting the options to the commands directly, it is also possible to set them globally using the `\rpgiconsset` command. Globally set options are overridden by options that are set directly.

```
\rpgiconsset{
    color=blue
}

\ability{charisma}
\ability{charisma}[color=red]
\ability{charisma}
```



```
style set={‹options›}
style add={‹options›}
```

To simplify styling, custom keys can be defined via the keys `style set` and `style add`. Both keys accept as value a key-value list consisting of one or multiple custom keys whose relevant value consists of another key-value list describing the relevant style. It is possible to use `#1` as placeholder for one argument. It is also possible to reference to an already define custom key in the definition of another custom key.

The key `style set` defines custom keys and sets their values. The key `style add` appends the relevant key-value list to the existing custom key as defined by `style set`. Note that it is not checked whether a custom key already exists and its definition will be overwritten without a warning.



```
\rpgiconsset{
    style set={
        foo={rotate=45, fill=blue},
        bar={foo, fill=red, scale=#1}
    }
}

\ability[negative]{charisma}[foo]
\class[negative]{warlock}[bar=2]
```

Note that the key `style appemd` has been deprecated and the key `style add` should be used instead.

alias={‹string›}{‹string›}

The key `alias` accepts exactly two (braced) arguments that represent names of shapes. A shape with the name given in the first argument is created as an exact copy of the shape with the name given in the second argument. If a shape with the name given in the first argument already exists, it will become a copy of the other shape without warning.

before sep={‹dimension›}
after sep={‹dimension›}
baseline={‹dimension›}

The spacing before and after the icons can be set using the options `before sep` and `after sep`. The option `baseline` can be used to adjust the baseline of the icons. These options can also be applied to the icon commands directly. The default value of `before sep` and `after sep` is 0.05 em. The default value of `baseline` is –3.5 pt.

Roll a die!
Roll a die!

```
Roll\die{eightside}{}a die!

\rpgiconsset{
    before sep={1cm}
}
Roll\die{eightside}{}a die!
```

actualtext={‹string›}

The option `actualtext` can be used to change the default replacement text that is used for example by screen readers in a tagged PDF. It expects as string that represents the relevant replacement text.

It is possible to change the replacement texts for specific icon types. For example, by setting `every saving={every charisma={actualtext={charisma saving}}}`, the replacement text for every instance of `\saving{charisma}` will be set to `charisma saving`.

Note that the possibility to provide a key-value list to `actualtext` taking shape names as keys and the relevant replacement texts as values has been removed. Instead, use the keys to set the style for all macros or shapes as shown in the previous paragraph.

4.4 Precompose icons

A lot of icons in a document can affect compilation time. If one or several icons are used several times, compilation time can be increased by precomposing these icons. In the precomposing process, the relevant icons are compiled into an external PDF file from which they are then loaded into

the document. This not only decreases compilation time, because every icon is only created once, but it also reduces the file size, because every icon is only embedded into the file once and further uses just reference to the first embedded instance.

In order to be able to compile an external PDF file during compilation (which required LaTeX to call another instance of LaTeX), the compilation needs to run with enabled shell escape.

In order to be able to include the external PDF file into the document, a backend that supports PDF inclusion (such as PDFLaTeX, LuaLaTeX or XeLaTeX) needs to be used.

```
\RPGIconsPrecomposeIcon[‹options›][‹reference›]{‹shape›}
\RPGIconsPrecomposeIcon*[‹options›][‹reference›]{‹shape›}
```

Using the macro `\RPGIconsPrecomposeIcon` icons can be precomposed. This macro should be used in the preamble of the document. It takes one mandatory argument that expects as string the name of the shape of the icon. The first optional argument can be used to provide options for styling the icon. The second optional argument can be used to define a reference for this comcrete styled icon. If no reference is defined, the name of the shape is used as reference. The starred version of the command is used to fill a frame with color instead of drawing its outline (see the macros `\RPGIconsUseIcon` and `\RPGIconsUseIcon*` for comparison).

For example, to precompose an icon that uses the `charisma` shape in black and another icon with the `charisma` shape in red with the reference `charisma-red`, the following two lines can be placed in the preamble of the document:

```
\RPGIconsPrecomposeIcon[scale=.333]{charisma}
\RPGIconsPrecomposeIcon[scale=.333, color=red][charisma-red]{charisma}
```

```
precompose
precompose={‹string›}
```

Once an icon has been precomposed, it can be called by adding the `precompose` option to the relevant macro. For example, `\ability{charisma}[precompose]` would load the shape from the external PDF that has been generated using the shape name as reference. To call an icon using a specific reference, this reference can be given as value to the `precompose` option. For example, the above described red variant of the `\ability{charisma}` shape could be called using `\ability{charisma}[precomppse=charisma-red]`.

Due to the fact that options that set colors, transformations or other styles have already been applied when the external PDF was generated, they won't have any effect when used together with the option `precompose`.

4.5 Roll dice syntax

```
\roll{‹roll syntax›}
\RPGIconsRoll{‹roll syntax›}
```

The `\roll` macro can be used to quickly typeset dice rolls with the relevant icons using the established dice rolling syntax. This syntax consists of a sequence of dice and numbers concatenated by mathematical operators (plus, minus or times). Typically, the letter `d` is used to denote a die with a certain number of sides. For example `d6` denotes a six-sided die. A number can be added to specify the number of such dice that are rolled together. The letter to denote the die can be changed using the option `roll syntax`.

For example, `2d6 + 3d4 - 1` means "roll two six-sided dice and three four-sided dice and subtract one from the result". The command `\roll{2d6 + 3d4 - 1}` results in 2 □ + 3 △ − 1.

The die notations `d2`, `d4`, `d6`, `d8`, `d10`, `d12`, `d20` and `d100` are defined. To denote a fudge die, `dF` can be used. To denote that the lowest or highest die should be removed from the result, the letters `L` and `H` can be used. The syntax `2d6 x 2` or `2d6 * 2` can be used to denote several rolls with the same set of dice.

If the `rpgicons` package is to be loaded together with some other package that defines the command `\roll`, the command `\RPGIconsRoll` can be used. This alternative command is an exact copy of the `\roll` command.

`roll syntax``={‹comma-separated list›}`

The option `roll syntax` can be used to change the character that denotes a die in the dice rolling syntax. Multiple characters can be given using a comma separated list. The default setting is `d,D` which allows notations such as `2d6` or `2D6`.

With `\rpgiconsset{roll syntax={w,W}}`, for example, notations such as `2w6` or `2W6` could be used.

5 Specifics of the PGF package variant

The PGF variant of the package is loaded by either calling `\usepackage[pgf]{rpgicons}` or `\usepackage{rpgicons-pgf}` in the preamble of the document after having installed the files `rpgicons.sty` and `rpgicons-pgf.sty`.

Since the commands to typeset the icons with the PGF variant of the package use `tikzpicture` environments, these commands should not be used inside another `tikzpicture`. However, because the package defines the icons as Ti*k*Z shapes, it is possible to use the icons in `tikzpicture` environments directly.

Apart from that, the PGF variant of the package provides a way to define custom commands to typeset the icons as boxed material which is safe to use in a `tikzpicture` context. Furthermore, the icons can be used as Ti*k*Z pics.

Once loaded, the PGF variant of the package defines a set of node shapes that can be used inside a `tikzpicture` environment.

`pics`

The PGF variant of the package provides the option `pics`. If the package is loaded with this option, every icon is also available as Ti*k*Z pic. On the use of pics, see section 5.6 below.

5.1 Icon commands

`rpg icons/`**`variant`**
`rpg icons/`**`variant`**`={‹integer›}`

The main user commands (as well as the underlying commands on which these commands are based as described below) accept as option the key `rpg icons/variant` which accepts an integer as value to select variants for the relevant shape. See the icon lists above for the respective variants. For example, the command `\ability{dexterity}[rpg icons/variant=1]` results in .

Setting the key `rpg icons/variant` without value is equivalent to setting its value to `1`. Selecting a non-existent variant selects the default shape.

With the following setting, all occurrences of the `necrotic` shape are set to the relevant variant.

```
\tikzset{
    rpg icons/every necrotic/.append style={
        rpg icons/variant=1
    }
}
```

```
\rpgiconsdie[‹style›]{‹shape›}[‹options›]{‹integer›}
\rpgiconsability[‹style›]{‹shape›}[‹options›]
\rpgiconssaving[‹style›]{‹shape›}[‹options›]
\rpgiconsspell{‹shape›}[‹options›]
\rpgiconsspellschool[‹style›]{‹shape›}[‹options›]
\rpgiconsdamage{‹shape›}[‹options›]
\rpgiconsattack{‹shape›}[‹options›]
\rpgiconscondition{‹shape›}[‹options›]
\rpgiconsclass[‹style›]{‹shape›}[‹options›]
\rpgiconsalignment[‹style›]{‹shape›}[‹options›]
\rpgiconscurrency{‹shape›}[‹options›]
\rpgiconsdirection[‹style›]{‹shape›}[‹options›]{‹integer›}
```

This set of commands are the basic commands defined by the PGF variant of the package on which the commands `\die`, `\ability`, `\saving`, `\spell`, `\spellschool`, `\damage`, `\attack`, `\condition`, `\class`, `\alignment`, `\currency` as well as `\direction` are based. This set of commands can be used in cases where another package defines one of these user commands. These user commands are exact copies of this set of commands.

5.2 Icon styles

Using the PGF variant of the package, all icons can by styled using arbitrary Ti*k*Z styles in general. As an example, `\die{eightside}[blue, thick]{2}` results in 2 .

If PDF tagging is activated by using `\DocumentMetadata{tagging=on}`, a replacement text is automatically added to the relevant icon which can be copied to the clipboard and can be read by screen readers. Note that tagging support depends on the used PDF reader. It is possible to add a replacement text to an icon using the `actualtext` key.

```
rpg icons/every die/.style={‹options›}
rpg icons/every ability/.style={‹options›}
rpg icons/every saving/.style={‹options›}
rpg icons/every spell/.style={‹options›}
rpg icons/every spellschool/.style={‹options›}
rpg icons/every damage/.style={‹options›}
rpg icons/every attack/.style={‹options›}
rpg icons/every condition/.style={‹options›}
rpg icons/every class/.style={‹options›}
rpg icons/every alignment/.style={‹options›}
rpg icons/every currency/.style={‹options›}
rpg icons/every direction/.style={‹options›}
rpg icons/every ‹shape›/.style={‹options›}
```

Using Ti*k*Z styles, all instances of a certain command or a certain shape can be styled at once. These styles all follow the pattern `rpg icons/every` followed by a space and the name of the command or the shape. For example, `\tikzset{rpg icons/every die/.append style={red}}` can be used to draw in red all icons created using the `\die` command. To draw every instance of the

`charisma` shape in red, `\tikzset{rpg icons/every charisma/.append style={red}}` can be used.

5.3 Setting styles globally

rpg icons/.style={‹options›}

All icons share the TikZ style `rpg icons` that is empty per default but can be used to style all icons at once. For example, if `\tikzset{rpg icons/.append style={draw=red}}` is placed at the beginning of the document, all icons will be drawn in red. Per default, the icons are drawn in the color of the surrounding text. Per default, the TikZ keys `line cap` and `line join` are set to `round`.

Note that it may be necessary to add the TikZ option `transform shape` when applying transformations to the icons, because the icons are constructed as TikZ nodes which are not affected by some transformations per default.

rpg icons/**background**={‹color›}

Some icons can be used with a negative color scheme where the icon is drawn negatively inside a filled shape. Per default, the icons are drawn in white in such cases, but it might be desirable that the icons are in the same color as the background. To this end, the color can be changed using the TikZ option `rpg icons/background` in the following way:



```
\colorbox{blue!50}{%
    \ability[negative]{charisma}
        [scale=2, transform shape]%
}

\tikzset{
    rpg icons/background={blue!50}
}
\colorbox{blue!50}{%
    \ability[negative]{charisma}
        [scale=2, transform shape]%
}
```

This feature can, of course, also be used to change the color of the icon independently from the color of the background.

Note that the key `rpg icons/background color` has been deprecated and the key `rpg icons/background` should be used instead.

rpg icons/**before sep**={‹dimension›}
rpg icons/**after sep**={‹dimension›}
rpg icons/**baseline**={‹dimension›}

The TikZ options `rpg icons/before sep` and `rpg icons/after sep` are used to define the width of the space that is added before and after the icons respectively. The default value of both lengths is 0.05 em. For example, setting the space before icons to 1 cm can be achieved as follows:

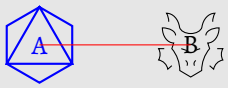
```
                                Roll\die{eightside}{}a die!

Roll a die!                     \tikzset{
Roll      a die!                    rpg icons/before sep={1cm}
                                }
                                Roll\die{eightside}{}a die!
```

The option `baseline` can be used to adjust the baseline of the icons. A larger value for the baseline will shift the icon downwards relative to the baseline of the surrounding text. The default value of the baseline is –3.5 pt.

5.4 Direct use of shapes

Because the icons are defined as Ti*k*Z shapes, they can directly be applied to Ti*k*Z nodes. However, the shapes don't have a shape border and no anchors defined, except for the `center` anchor that sits exactly in the center of the shape. Therefore, if nodes with these shapes are connected using edges, the `center` anchor will be used to connect the nodes. If nodes with these shapes are being positioned, only the `center` anchor is available. Text content of these nodes is simply printed on top of the center of the node. Compare the following example.




```
\begin{tikzpicture}
    \node[eightside, draw, blue, thick]
        at (0,0) (A) {A};
    \node[charisma, draw]
        at (2,0) (B) {B};
    \draw[red] (A) -- (B);
\end{tikzpicture}
```

The variant shapes are accessible by adding the variant number as small roman numeral to the shape name. For example, to access the second variant of the `druid` shape, use `druid ii`. This is also possible for boxed icons and icons as pics as described below.

5.5 Boxing of icons

Because the icons cannot simply be used inside `tikzpicture` environments, the PGF variant of the package provides a workaround to place icons inside boxes for later use. Icons that are boxed this way can safely used inside `tikzpicture` environments. This might be necessary, if an icon should be used in inline text that sits inside a node.

`\provideprotectedrpgicon``{‹command›}[‹style›]{‹shape›}[‹options›]{‹box name›}`

The command `\provideprotectedrpgicon` creates a box containing the icon that would be created using one of the regular commands this package provides.

`\provideprotectedrpgicon{die}[large]{eightside}[blue, thick]{mybox}`, for example, stores the icon of an eight-sided die with the relevant style and Ti*k*Z options in a new box named `mybox`. Note that no integer can be added to the macros `die`, `currency` or `direction` in this context.

`\useprotectedrpgicon``{‹box name›}`

Using the command `\useprotectedrpgicon`, the previously defined box can be used to place the relevant icon. With the above definition, `\useprotectedrpgicon{mybox}` would result in .

Having created a boxed icon, it is safe to use it, for example, inside a TikZ node:




```
\begin{tikzpicture}
    \node[circle, draw, align=center] {
        \useprotectedrpgicon{mybox} \\
        Roll a die!
    };
\end{tikzpicture}
```

5.6 Icons as pics

If the PGF variant of the package is loaded with the option `pics`, every icon is also available as TikZ pic. The names of the pics always start with `rpg icons` followed by a space and the name of the relevant icon (see the lists above). For abilities, savings, spellschools, damages, classes and alignments, additional pics containing the relevant frame exist where the name has the suffixes `ability`, `saving`, `spellschool`, `damage`, `class`, and `alignment` respectively.

The icon is embedded as a node in the pic which has the name `-node`. Thus, it is possible to name the pic and refer to the node inside. Due to the fact that the icon is a node, the option `transform shape` has to be used if transformations on the pic should affect the node as well. It is possible to apply styles to the node using the TikZ option `every node` as shown in the following example.



```
\begin{tikzpicture}
    \pic[
        transform shape,
        scale=2,
        fill=blue,
        draw=red,
        every node/.append style={
            white,
            thick
        }
    ] (p) {rpg icons charisma ability};
    \draw[red] (p-node) -- +(2,0);
\end{tikzpicture}
```

```
rpg icons/create pic from shape={‹string›}
rpg icons/create pic from ability shape={‹string›}
rpg icons/create pic from saving shape={‹string›}
rpg icons/create pic from spellschool shape={‹string›}
rpg icons/create pic from damage shape={‹string›}
rpg icons/create pic from class shape={‹string›}
rpg icons/create pic from alignment shape={‹string›}
rpg icons/create every style={‹string›}
```

The PGF variant of the package defines five TikZ keys that are used to create pics using the relevant node shapes. Another key is defined to create keys that can be used to style all instances of a command or shape. In normal circumstances, it is not necessary to use these keys. They are mentioned here only for reference. These keys expect as value a string representing the relevnat shape name. It is possible to use the `/.list` handler provided by PGF to apply the key to multiple values at once.

The following example shows how to create the drawing on the first page of this documentation using TikZ pics. We need to call `rpg icons/create pic from ability shape` for the

shapes `twentyside`, `buff` and `ranged`, because per default no pics are defined for these shapes in combination with ability. Note the use of `\space` to ensure correct use of spaces in the pic name as spaces are gobbled after commands in TeX.



```
\tikzset{
    rpg icons%
    /create pic from ability shape%
    /.list={
        twentyside,
        buff,
        ranged
    },
    rpg icons/every ability/.style={
        ultra thick,
        draw=white,
        line join=round,
        line cap=round
    }
}

\begin{tikzpicture}[scale=4]
    \foreach \x/\c [count=\i] in {
        twentyside/264653,
        charisma/287271,
        armour/2a9d8f,
        buff/e9c46a,
        ranged/f4a261,
        proficiency/e76f51
    } {
        \definecolor{color}{HTML}{\c}
        \pic[
            fill=color,
            draw=none,
            transform shape
        ] at ({60*\i+10}:{0.33cm})
            {rpg icons \x\space ability};
    }
\end{tikzpicture}
```

5.7 Roll dice syntax

```
\roll{‹roll syntax›}
\rpgiconsroll{‹roll syntax›}
```

Please refer to section 4.5 about how to use the `\roll` macro in general. Differences to the L3 variant of the package are described in the following.

The letter to denote the die can be changed using the TikZ style `rpg icons/roll syntax`.

If the `rpgicons` package is to be loaded together with some other package that defines the command `\roll`, the command `\rpgiconsroll` can be used. This alternative command is an exact copy of the `\roll` command.

```
rpg icons/roll syntax={‹comma-separated list›}
```

The TikZ style `rpg icons/roll syntax` can be used to change the character that denotes a die in the dice rolling syntax. Multiple characters can be given using a comma separated list. The default setting is `d,D`.

6 L3 token variables and commands

The L3 variant of the package provides the following token variables and commands for defining user functions by package authors.

\rpgicons_draw_shape_⟨shape⟩:

These commands contain the definitions of the shapes of the icons. If the shape name contains a space, this is substituted by an underscore in the command name. For example, the command `\rpgicons_draw_shape_charisma:` contains the definition of the charisma shape, the command `\rpgicons_draw_shape_psion_i:` contains the definition of variant one of the psion shape (`psion i`).

The definition of a shape consists of a row of commands to describe one or more paths. The definition should be a valid code snippet that can be placed between the commands `\draw_begin:` and `\draw_path_use:n { stroke } \draw_end:`. The shape `lawful neutral`, for example, has the following definition:

```
\cs_new_protected:Npn \rpgicons_draw_shape_lawful_neutral: {
    \draw_path_circle:nn { 0cm , 0cm } { .2cm }
    \draw_path_moveto:n { .4cm , 0cm }
    \draw_path_lineto:n { -.4cm , 0cm }
}
```

\rpgicons_draw_shape_sixside_pips:n {⟨integer⟩}

The command `\rpgicons_draw_shape_sixside_pips:n` takes as argument an integer from one to nine and outputs the definition of paths representing the relevant number and arrangement of pips inside the shape of a sixside die.

\rpgicons_draw_frame_ability:
\rpgicons_draw_frame_saving:
\rpgicons_draw_frame_spellschool:
\rpgicons_draw_frame_damage:
\rpgicons_draw_frame_class:
\rpgicons_draw_frame_alignment:

These commands contain the definitions of the relevant frames that are used by the relevant icon commands.

\l_rpgicons_icon_before_sep_dim
\l_rpgicons_icon_after_sep_dim
\l_rpgicons_icon_baseline_dim
\l_rpgicons_icon_linewidth_dim

```
\l_rpgicons_icon_frame_tl
\l_rpgicons_icon_color_stroke_tl
\l_rpgicons_icon_color_fill_tl
\l_rpgicons_icon_color_text_tl
\l_rpgicons_icon_color_background_tl
```

```
\l_rpgicons_icon_variant_int
```

```
\l_rpgicons_icon_opacity_stroke_fp
\l_rpgicons_icon_opacity_fill_fp
\l_rpgicons_icon_opacity_text_fp
\l_rpgicons_icon_opacity_background_fp
\l_rpgicons_icon_transform_scale_fp
\l_rpgicons_icon_transform_scale_inner_fp
\l_rpgicons_icon_transform_rotate_fp
```

```
\l_rpgicons_icon_actualtext_str
```

These token lists contain the values set by the ueer via the relevant options. The floating point token lists `\l_rpgicons_icon_opacity_stroke_fp`, `\l_rpgicons_icon_opacity_fill_fp`, `\l_rpgicons_icon_opacity_text_fp`, `\l_rpgicons_icon_opacity_background_fp`, as well as `\l_rpgicons_icon_transform_scale_fp` and `\l_rpgicons_icon_transform_scale_inner_fp` are initialised with the value 1.0.

```
\l_rpgicons_icon_precompose_ref_str
\l_rpgicons_precompose_output_str
```

The token list `\l_rpgicons_icon_precompose_ref_str` contain the values set by the ueer via the option `precompose`.

The token list `\l_rpgicons_precompose_output_str` contains the value set by the user via the option `precompose output`. Its default value is `\c_sys_jobname_str _rpgicons` which is the name of the current file with the suffix `_rpgicons` added.

```
\rpgicons_shape_set_eq:nn {‹shape›} {‹shape›}
```

The command `\rpgicons_shape_set_eq:nn` sets the definition of the shape given in the first argument equal to that given in the second argument. If either shape name contains spaces, these are replaced by underscores. The relevant internal command is then set to contain the relevant shape definition. For example, `\rpgicons_shape_set_eq:nn {foo} {charisma}` will create the command `\rpgicons_draw_shape_foo:` as copy of `\rpgicons_draw_shape_charisma:`. If the shape given in the first argument already exits, its definition will be replaced.

7 Changes

v1.1.0 (2023/08/15) First public release.

v1.1.1 (2023/11/15) Fudge dice icon added.

v1.1.2 (2023/11/16) Bug fixed that caused wrong spacing when using dice icons without quantifier.

v1.2.0 (2023/11/20) Corrections in the manual. Icons for six-sided dice with one to nine pips, plus sign and minus sign added.

v1.3.0 (2023/11/21) Option to set background color added. Renamed global option.

v1.3.1 (2024/02/18) Correction of initializing code. Correction of default value of after sep. Addition of pics.

v1.4.0 (2024/02/21) L3 variant added.

v1.4.2 (2024/02/21) Alternative set of commands in L3 variant defined.

v1.4.4 (2024/02/24) Added styles for every instance of command or shape, implementation of recent changes to `l3draw` code.

v1.5.0 (2024/02/25) Alternative set of commands defined, added support of styles in pics.

v1.5.1 (2024/02/28) Addition of opacity to L3 variant.

v1.5.4 (2024/03/06) Correction of baseline settings in L3 variant, added accessibililty support for L3 variant.

v1.6.0 (2024/03/15) Four attribute icons added, minor correction of styles.

v1.6.1 (2024/03/16) Unified size of negative attribute icon.

v1.7.0 (2024/03/16) Macro for easy typesetting using roll dice syntax added in L3 variant, compatibility mode updated.

v1.8.0 (2024/03/24) Unified wrapper to load either package variant.

v1.8.2 (2024/04/28) Roll dice syntax for PGF variant.

v1.8.4 (2024/08/08) Bug fix in evocation spellschool shape. Unification of frames in PGF and L3 variants.

v1.8.8 (2025/06/15) Adjustments to support next release of `l3draw` package.

v1.9.0 (2025/06/30) Improve check for PDF management, deprecate compatibility mode, improve accessibility support.

v2.0.0 (2025/09/25) Change accessibility support to automatic tagging support.

v2.0.4 (2025/10/21) Added key to append styles for every shape or type.

v2.1.0 (2025/10/27) Added alignment and currency shapes.

v2.2.0 (2025/11/05) Added shapes for ritual spell, gems and jewellery as well as for classes. Fixes in alignment shapes and in pic definitions.

v2.2.1 (2025/11/06) Added alternative shapes for cleric, druid, sorcerer and necrotic. Update of class and alignment frames. Fixes in transformation keys in L3 variant.

v2.3.0 (2025/11/16) Added key to define custom keys in L3 variant. Added shape for emanation.

v2.3.3 (2025/12/15) Added variant for wisdom shape.

v2.4.0 (2026/01/15) Added key to access variants for class shapes.

v2.4.1 (2026/01/21) Added variant for psion shape.

v2.5.0 (2026/01/27) Added variants for paladin, monk and wizard shapes. Standardization of key names.

v2.5.1 (2026/02/06) Added key to generate alias for shapes in L3 variant. Update documentation.

v2.5.2 (2026/02/15) Added variant for melee, bard and jewellery shape. Update documentation.

v2.6.0 (2026/02/25) Added functionality to precompose icons in L3 variant. Update accessibility support.

v2.7.0 (2026/03/30) Added default replacement text provided if tagging is active in PGF variant. Added alias names for dice shapes.

v2.8.0 (2026/04/15) Added movement shapes.

v2.9.0 (2026/04/30) Added line cap and line join options for L3 variant.