The `rpgicons` package

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

Jasper Habicht [*]

Version 2.3.0, released on 11 November 2025

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` 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 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`. Alternatively, `\usepackage{rpgicons-l3}` can be called instead.

```
pgf
```

To load the PGF variant of the package the package needs to be loaded with the option `pgf`. Alternatively, `\usepackage{rpgicons-pgf}` can be called instead.

[*] 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 and to all other contributors to this package.

2.1 Dependencies

The L3 variant of the package loads the `l3draw` package.

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.

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. Additionally, 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 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. 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 ⬡2. 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
`\die`	⬭	`twoside`
	△	`fourside`
	□	`sixside`
	⬡	`eightside`
	◇	`tenside`
	⬠	`twelveside`
	⬡	`twentyside`
	◔	`hundredside`
	±	`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
`\ability`	🐻	`strength`

Command	Icon	Shape
		`dexterity`
		`dexterity alt`
		`constitution`
		`intelligence`
		`wisdom`
		`charisma`
		`resilience`
		`sanity`
		`perception`
		`luck`
		`armor`
		`proficiency`
`\saving`		`strength`
		`dexterity`
		`dexterity alt`
		`constitution`
		`intelligence`
		`wisdom`
		`charisma`
		`resilience`
		`sanity`
		`perception`
		`luck`
		`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
`\spell`		`linear`
		`conic`
		`quadratic`
		`cubic`
		`spheric`
		`cylindric`
		`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
`\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
`\damage`	○	`acid`
	○	`bludgeoning`
	○	`cold`
	○	`fire`
	○	`force`
	○	`lightning`
	○	`necrotic`
	○	`necrotic alt`
	○	`piercing`
	○	`poison`
	○	`psychic`

Command	Icon	Shape
	◎	`radiant`
	⊗	`slashing`
	◯	`thunder`
	♡	`healing`
`\attack`	⚔	`melee`
	✧	`ranged`
	✦	`magic`
	✋	`singlehanded`
	✋	`doublehanded`
`\condition`	✋	`buff`
	⊘	`blinded`
	☺	`charmed`
	⊘	`deafened`
	☹	`exhausted`
	☹	`frightened`
	✊	`grappled`
	✲	`incapacitated`
	◌	`invisible`
	☹	`paralyzed`
	☹	`petrified`
	☹	`poisoned`
	⊃	`prone`
	♛	`restrained`
	☹	`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
`\class`	▣	`artificer`
	▣	`barbarian`

Command	**Icon**	**Shape**
	▣	`bard`
	▣	`cleric`
	▣	`cleric alt`
	▣	`druid`
	▣	`druid alt`
	▣	`fighter`
	▣	`monk`
	▣	`paladin`
	▣	`ranger`
	▣	`rogue`
	▣	`sorcerer`
	▣	`sorcerer alt`
	▣	`warlock`
	▣	`wizard`
`\alignment`	⬡	`lawful good`
	⬡	`neutral good`
	⬡	`chaotic good`
	⬡	`lawful neutral`
	⬡	`true neutral`
	⬡	`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**
`\currency`	⊙	`copper`
	△	`silver`
	⧗	`gold`
	⊙	`electrum`
	◇	`platinum`
	💎	`gem`
	♉	`jewellery` or `jewelry`

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.

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

4.1 Icon commands

```
\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` and `\currency` are based on this command.

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 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›]
```

The L3 variant of the package defines a set of commands on which the user commands `\die`, `\ability`, `\saving`, `\spell`, `\spellschool`, `\damage`, `\attack`, `\condition`, `\class`, `\alignment` as well as `\currency` 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

```
frame
stroke
fill
text
color
background
stroke opacity
fill opacity
text opacity
opacity
background opacity
line width
scale
scale inner
rotate
```

The `\RPGIconsUseIcon` command and the commands `\die`, `\ability`, `\saving`, `\spell`, `\spellschool`, `\damage`, `\attack`, `\condition`, `\class`, `\alignment` and `\currency` 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.

With the `frame` option, one of four 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 commands `\ability`, `\saving`, `\spellschool` and `\damage` make use of the relevant frame.

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` or the `\spellschool` command or using the starred version of the `\RPGIconsUseIcon` command.

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. The option `line width` sets the line width for strokes. Using the `scale` and `rotate` options, the shape can be scaled and rotated.

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

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
every ability
every saving
every spell
every spellschool
every damage
every attack
every condition
every class
every alignment
every currency
every ‹shape›
```

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
every ability add
every saving add
every spell add
every spellschool add
every damage add
every attack add
every condition add
every class add
every alignment add
every currency add
every ‹shape› add
```

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
style append
```

To simplify styling, custom keys can be defined via the keys `style set` and `style append`. 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 append` 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]
```

```
before sep
after sep
baseline
```

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
```

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 value a key-value list where the key is the name of the icon shape and the value is the relevant replacement text. For example, with setting `actualtext={sixside={d6}}`, a screen reader would read "d six" for icons using the `sixside` shape.

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

4.4 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
```

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

```
\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›]
```

The PGF variant of the package defines a set of commands on which the user commands `\die`, `\ability`, `\saving`, `\spell`, `\spellschool`, `\damage`, `\attack`, `\condition`, `\class`, `\alignment` as well as `\currency` 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}`, it is possible to add a replacement text to an icon using the `actualtext` key.

```
rpg icons/every die
rpg icons/every ability
rpg icons/every saving
rpg icons/every spell
rpg icons/every spellschool
rpg icons/every damage
rpg icons/every attack
rpg icons/every condition
rpg icons/every class
rpg icons/every alignment
rpg icons/every currency
rpg icons/every ‹shape›
```

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

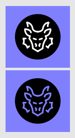
```
rpg icons
```

All icons share the Ti*k*Z 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.

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 color` in the following way:



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

\tikzset{
    rpg icons/background color={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.

```
rpg icons/before sep
rpg icons/after sep
rpg icons/baseline
```

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⬡a die!
Roll   ⬡a die!

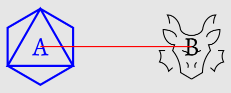
```
Roll\die{eightside}{}a die!

\tikzset{
    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 TikZ shapes, they can directly be applied to TikZ 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}
```

5.5 Boxing of icons

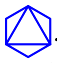
Because the icons cannot simply be used inside `tikzpicture` environments, the PGF variant of the package provides a workaround to place icons inside boxes for later use. Icons that are boxed this way can safely used inside `tikzpicture` environments. This might be necessary, if an icon should be used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

`\provideprotectedrpgicon``{⟨command⟩}[⟨style⟩]{⟨shape⟩}[⟨options⟩]{⟨box name⟩}`

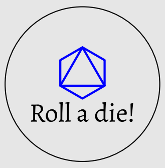
The command `\provideprotectedrpgicon` creates a box containing the icon that would be created using one of the regular commands this package provides.

`\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 `die` command 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 Ti*k*Z 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 Ti*k*Z 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 and damages, additional pics exist where the name has the suffixes `ability`, `saving`, `spellschool`, and `damage` 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 Ti*k*Z 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
rpg icons/create pic from ability shape
rpg icons/create pic from saving shape
rpg icons/create pic from spellschool shape
rpg icons/create pic from damage shape
rpg icons/create pic from class shape
rpg icons/create pic from alignment shape
rpg icons/create every style
```

The PGF variant of the package defines five Ti*k*Z 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.

The following example shows how to create the drawing on the first page of this documentation using Ti*k*Z 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,
        armor/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 4.4 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 Ti*k*Z 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
```

The Ti*k*Z 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 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 (2025/11/11) Added key to define custom keys in L3 variant.