

Rubikrotation examples¹

NOTE: Since the `\RubikRotation` command calls the perl script `rubikrotation.pl` this `.tex` file needs to be run using the `--shell-escape` command-line switch, as follows:

```
latex --shell-escape example.tex
```

If you forget to use the command-line switch, the file will run OK, but the cubes will remain in the initial ‘solved’ configuration. The `tikz` package must be loaded before the two `rubik..` packages.

Example 1

In Figure 1 we show the so-called “sixspot” configuration, generated from a solved cube using the rotation sequence **U, D’, R, L’, F, B’, U, D’** (see the website of Reid at www.cflmath.edu/~reid/Rubik/patterns.html).

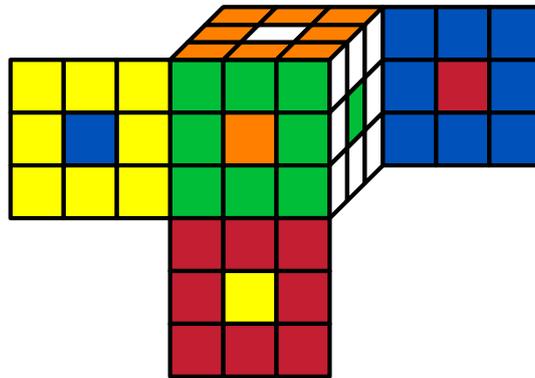


Figure 1: The so-called ‘sixspot’ configuration (Reid).

Creating a macro to hold a rotation sequence greatly facilitates their use, as follows:

```
\newcommand{\sixspot}{[sixspot],U,Dp,R,Lp,F,Bp,U,Dp}
```

We can now process this sequence using its macro name as an argument for the `\RubikRotation` command. The code used for the above Figure uses the `\ShowCube{}{}{}` command for which #1 is the minipage width, #2 is the `tikzpicture` scale factor (0–1), and #3 is a `\Draw..` command (see RUBIKROTATION documentation for details).

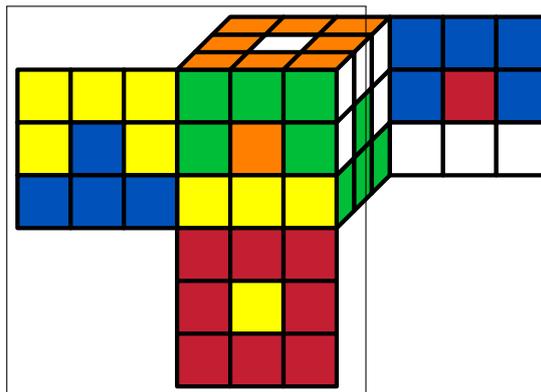
```
\usepackage{tikz,rubikcube,rubikrotation}
\thispagestyle{empty} %% disables pagenumbers
...
\begin{figure}[hbt]
  \centering
  \RubikCubeSolved
  \RubikRotation{\sixspot}
  \ShowCube{7cm}{0.7}{\DrawRubikCubeFlat}
  \caption{...}
\end{figure}
```

Note that running the ‘sixspot’ sequence 3 times returns the cube to its original ‘solved’ state (it is a so-called ‘order 3’ sequence). The command for processing it three times is `\RubikRotation[3]{\sixspot}`.

¹These examples are part of the RUBIKROTATION package, and assume some familiarity with the Rubik bundle documentation `rubikcube.pdf` and `rubikrotation.pdf`.

Example 2

In this example we demonstrate the use of the `\ShowRubikErrors` command, which places a copy of the Perl output file `rubikstateERRORS.dat` underneath the graphic so you can see a list of all the errors, if any (in this example we have introduced a few errors—wrong minipage width, typos and some animals—into the rotation sequence). It is important to note that the `\ShowRubikErrors` command must be placed *after* the TikZ picture environment (i.e., in this case after the `\ShowCube` command). Note that full details of all errors are also included in the `.log` file.



```
%% rubikstateERRORS.dat
%% ---(RR.sty v3.0): comments output by Perl script
** ERROR: rotation,[sixspot],U,Dp,R,Lp,F,Bp,U,Dpppp,cat,dog
** ERROR: [Dpppp] in RubikRotation{}
** ERROR: [cat] in RubikRotation{}
** ERROR: [dog] in RubikRotation{}
```

Figure 2: The same ‘sixspot’ sequence of rotations as shown in Example 1, but now with some errors (wrong minipage-width, typos & animals!) in the rotation sequence (it should be just `U,Dp,R,Lp,F,Bp,U,Dp`).

In this example we have used the F version of the `\ShowCube` command (`\ShowCubeF`) which places an fbox around the image so you can see the extent of any white space etc. This reveals that the set `minipage-width` (4.5cm) in the `\ShowCubeF` command—see code below—is too narrow: it should be 7cm (10×0.7) to just include the whole image. Once fixed, we can remove the F from the `\ShowCubeF` command. Note also that only ‘`\Draw...`’ commands really need to be inside the TikZ picture environment (i.e., inside the `\ShowCube` command). The above figure was generated by the following code.

```
\RubikCubeSolved
\RubikRotation{[sixspot],U,Dp,R,Lp,F,Bp,U,Dpppp,cat,dog}
\begin{figure}[hbt]
  \centering
  \ShowCubeF{4.5cm}{0.7}{\DrawRubikCubeFlat}
  \ShowRubikErrors
  \caption{...}
\end{figure}
```

Even if the `\ShowRubikErrors` command is not used, it is always a good idea to check the file `rubikstateERRORS.dat` after a \LaTeX run, since this file will reveal any errors.

Example 3

In this example we use the `\RubikRotation` command to scramble a ‘solved’ Rubik cube via a sequence of 120 random rotations, using the following command (the details of the process can be seen in the `.log` file):

```
\RubikRotation{random,120}
```

In this example, we also save the final configuration (state) to a file (`exampfig4.tex`) using the command `\SaveRubikState{exampfig4.tex}` so we can display the cube in the same state later (see Example 4) but in a different format. Note that since we are using a random sequence, it follows that each time this file is run not only will a visually different cube be generated, but the same state will be shown in both Figures 3 and 4.

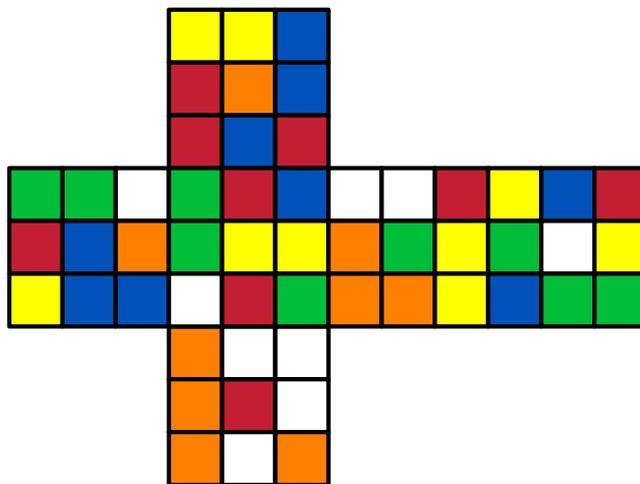


Figure 3: This shows a cube generated by 120 random rotations

```
\usepackage{tikz,rubikcube,rubikrotation}
...
\begin{figure}[hbt]
  \centering
  \RubikCubeSolved
  \RubikRotation{random,120}
  \SaveRubikState{exampfig4.tex}
  \ShowCube{8.4cm}{0.7}{\DrawRubikFlat}
\caption{...}
\end{figure}
```

Q: How do we determine the minipage-width (8.4cm) in the `\ShowCube` command?

A: The object is 12 cubie squares wide (1 cm each). Since the specified TikZ scale factor is 0.7 here, then the true width of the image is $12 \times 0.7 = 8.4$ cm. If you change the scale then the size of the image will change, and a new width will be required to just fit the image.

Note that in this particular case (only a single image in the ‘figure’ environment), since the `\ShowCube` command places the image centrally inside the minipage, the image will in fact be centrally placed in the `\textwidth` provided the image is *smaller* than the `fbox`—i.e., if we used instead a minipage-width of, say, 12 cm the image would still appear centered in the `\textwidth` in this case. However, when there are several images in the ‘figure’, then the spacing may appear strange unless each image closely fits its own minipage-width etc. It is often useful, therefore, to check the size of the `fbox` (using the `\ShowCubeF` command).

Example 4

In this example we display a cube having the same state as that shown in Figure 3. The configuration state was saved from Figure 3 using the command `\SaveRubikState{exampfig4.tex}`, and then input here using `\input{exampfig4.tex}`. These commands therefore allow the state of a previous cube to be saved to a file, and then displayed again later in a different format.

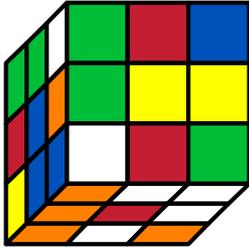


Figure 4: This shows a Rubik cube in exactly the same state as the one shown in Figure 3

```
\usepackage{tikz,rubikcube,rubikrotation}
...
\begin{figure}[hbt]
  \centering
  \input{exampfig4.tex}
  \Showcube{4cm}{0.8}{\DrawRubikCubeLD}
\caption{...}
\end{figure}
```

Example 5

Here we show a convenient way of displaying a series of small cubes showing a sequence of rotations (**U**, **R**, **F**).

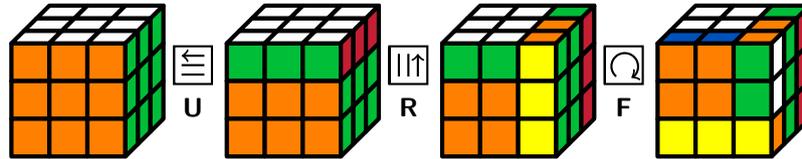


Figure 5: The rotations **U**, **R**, **F** on a solved cube.

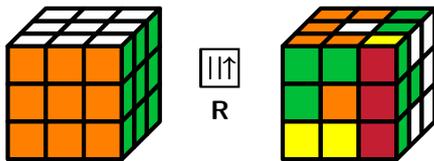
The code for the above sequence is as follows:

```
\usepackage{tikz,rubikcube,rubikrotation}
...
\begin{figure}[hbt]
\centering
\RubikCubeSolved
\ShowCube{2cm}{0.5}{\DrawRubikCubeRU}
\Rubik{U}
\RubikRotation{U}\ShowCube{2cm}{0.5}{\DrawRubikCubeRU}
\Rubik{R}
\RubikRotation{R}\ShowCube{2cm}{0.5}{\DrawRubikCubeRU}
\Rubik{F}
\RubikRotation{F}\ShowCube{2cm}{0.5}{\DrawRubikCubeRU}
\caption{The rotations \rr{U}, \rr{R}, \rr{F} on a solved cube.}
\end{figure}
```

Example 6

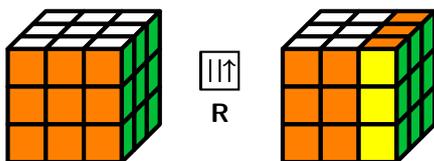
In this example we highlight the fact that commands used inside a TikZ picture environment remain local to that environment. Note that this applies with other environments too; for example, with both the `\minipage` the `\figure` environments. Consequently, it is generally best when drawing a sequence of cubes to reserve the TikZ picture environment (i.e., the `\ShowCube` command) only for Rubik `\Draw...` commands and TikZ commands.

In this example the first cube uses the `\RubikCubeSolved` *inside* the TikZ environment. However, if we then perform the rotation `R`  the command `RubikRotation{R}` results in something quite unexpected (and obviously not correct). This is because the effect of the `\RubikCubeSolved` command is not visible outside its TikZ environment, and hence the `RubikRotation{R}` command has to operate on the current globally available colour state information; i.e., that following the action of the `\RubikRotation{[sixspot],...}` command in the earlier Example 2 (shown in Figure 2) which was executed *before, and outside* the `figure` environment in that example, and hence the strange form of the second cube below.



```
\usepackage{tikz,rubikcube,rubikrotation}
...
\begin{minipage}{0.4\textwidth}
  \ShowCube{2cm}{0.5}{%
    \RubikCubeSolved
    \DrawRubikCubeRU
  }
  \hspace{4mm}\Rubik{R}\hspace{5mm}%
  \RubikRotation{R}%
  \ShowCube{2cm}{0.5}{\DrawRubikCubeRU}
\end{minipage}
```

If we now bring the `\RubikCubeSolved` command out and place it before the `\ShowCube` command then its ‘state’ information becomes globally accessible, and can therefore be acted upon (& updated) by the `\RubikRotation{R}` command, and hence is rendered (correctly for us) by the next `\DrawRubikCubeRU` command.



```
\begin{minipage}{0.4\textwidth}
  \RubikCubeSolved
  \ShowCube{2cm}{0.5}{\DrawRubikCubeRU}
  \hspace{4mm}\Rubik{R}\hspace{5mm}%
  \RubikRotation{R}%
  \ShowCube{2cm}{0.5}{\DrawRubikCubeRU}
\end{minipage}
```

Example 7

The ‘superflip’ configuration

Once you have mastered Rubik’s cube, then an interesting exercise is to generate the so-called ‘superflip’ configuration, in which all the corners are correctly solved, while all the edges are flipped.

For the impatient, the superflip sequence of 24 quarter-turn rotations (listed on Randelshofer’s website (www.randelshofer.ch/rubik/patterns/U080.01.html) is as follows. This converts the solved cube on the left into the configuration shown on the right.



Surprisingly, this sequence is actually equivalent to $\left\{ \left(\begin{array}{cc} \boxed{\downarrow} & \boxed{\rightleftharpoons} \\ \mathbf{M} & \mathbf{U}' \end{array} \right\}_4, [y], [x] \}_3$

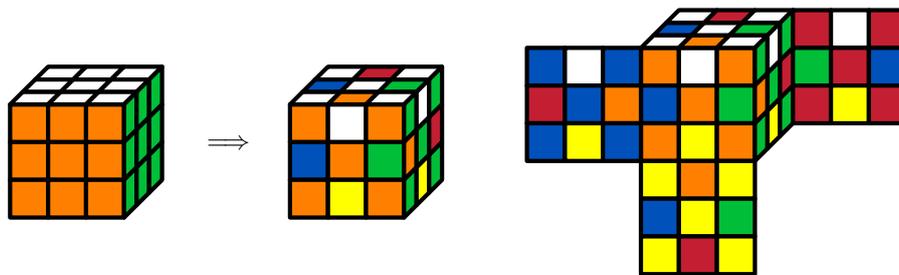


Figure 6: Two representations of the superflip configuration.

The full code for the figure above is as follows:

```
\usepackage{tikz,rubikcube,rubikrotation}
...
\begin{center}
\Rubik{U}\Rubik{R}\Rubik{R}\Rubik{Fp}\Rubik{R}\Rubik{Dp}\Rubik{L}\Rubik{Bp}%
\Rubik{R}\Rubik{Up}\Rubik{R}\Rubik{Up}\Rubik{D}\Rubik{Fp}\Rubik{U}\Rubik{Fp}%
\Rubik{Up}\Rubik{Dp}\Rubik{B}\Rubik{Lp}\Rubik{Fp}\Rubik{Bp}\Rubik{Dp}\Rubik{Lp}
\end{center}

\medskip

%% brace and bracket macros
\newcommand{\Rubikbracket}[1]{\left(\mbox{#1}\right)}
\newcommand{\Rubikbrace}[1]{\left\{\mbox{#1}\right\}}

{\noindent}Surprisingly, this sequence is actually equivalent to
\Rubikbrace{\Rubikbracket{\Rubik{M}\Rubik{Up}}}_4, \Rubik{yp}, \Rubik{x}}_3

\newcommand{\superflip}{U,R2,Fp,R,Dp,L,Bp,R,Up,R,Up,D,%
Fp,U,Fp,Up,Dp,B,Lp,Fp,Bp,Dp,Lp}

\begin{figure}[hbt]
\centering
```

```
\RubikCubeSolved%
\ShowCube{2cm}{0.5}{\DrawRubikCubeRU}
\hspace{5mm}$\Longrightarrow$\hspace{5mm}%
\RubikRotation{\superflip}%
\ShowCube{2cm}{0.5}{\DrawRubikCubeRU}
\hspace{1cm}%
\ShowCube{5cm}{0.5}{\DrawRubikCubeFlat}
\caption{...}
\end{figure}
```

— END —