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.math.ucf.edu/~reid/Rubik/`).

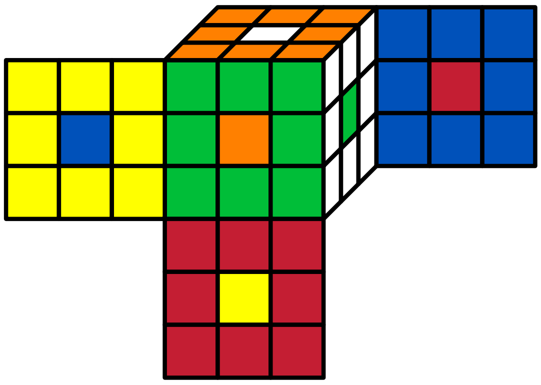


Figure 1: The so-called ‘sixspot’ configuration (Reid).

```
----------------
\documentclass[a4paper]{article}
\usepackage{tikz,rubikcube,rubikrotation}
\thispagestyle{empty} %% disables pagenumbers
\begin{document}
\begin{tikzpicture}[scale=0.9]
    \RubikCubeSolved
    \RubikRotation{*sixspot,U,Dp,R,Lp,F,Bp,U,Dp}
    \DrawRubikCubeFlat
\end{tikzpicture}
\end{document}
--------------
```

Note that a more flexible approach is to incorporate such rotation sequences into a ‘newcommand’. For example, in this case we can write

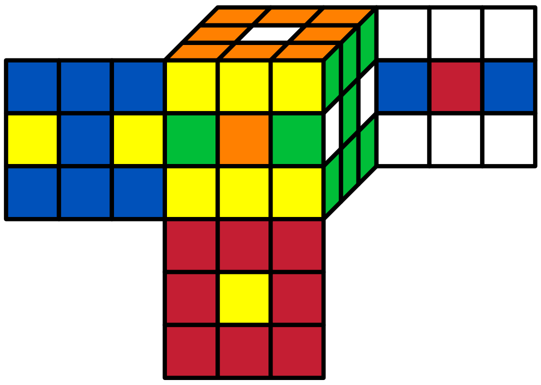
```
\newcommand{\sixspot}{*sixspot,U,Dp,R,Lp,F,Bp,U,Dp}
```

which allows the code to Figure 1 to be more easily written as:

```
\begin{tikzpicture}
    \RubikCubeSolved
    \RubikRotation{\sixspot}
    \DrawRubikCubeFlat
\end{tikzpicture}
```

Example 2

In this example we show 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 couple of errors). It is important to note that the `\ShowRubikErrors` command must be placed *after* the TikZ picture environment. Note also that full details of all errors are included in the .log file.



```
%% rubikstateERRORS.dat
** ERROR: rotation,*sixspot,U,Dp,R,Lp,F,Bp,UU,Dpp
** ERROR: [UU] in RubikRotation
** ERROR: [Dpp] in RubikRotation
```

Figure 2: The same 'sixspot' sequence of rotations as before, but now with some typo errors in the rotation sequence (should be `U,Dp,R,Lp,F,Bp,U,Dp`.

Note that only 'Draw' commands really need to be inside the TikZ picture environment—see the following code which we used for the above figure.

```
-----------------
\RubikCubeSolved
\RubikRotation{*sixspot,U,Dp,R,Lp,F,Bp,UU,Dpp}
%
\begin{tikzpicture}
   \DrawRubikCubeFlat
\end{tikzpicture}
\ShowRubikErrors
-----------------
```

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 (`exmpfig4.tex`) using the command `\SaveRubikState{exmpfig4.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.

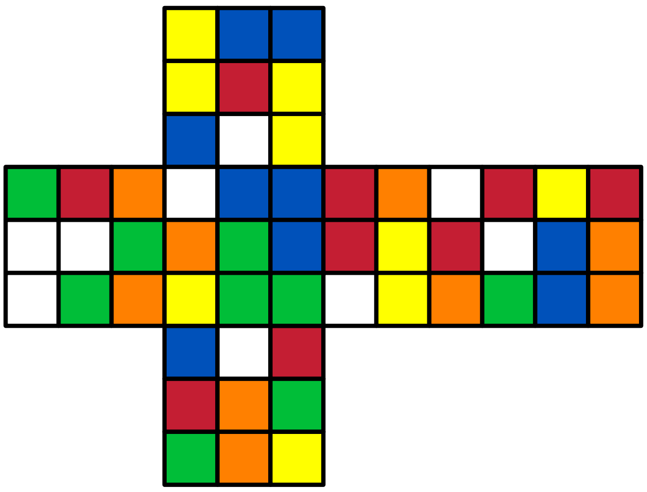


Figure 3: This shows a cube generated by 120 random rotations

```
--------------------
\begin{figure}[hbt]
\centering
\begin{tikzpicture}[scale=0.8]
    \RubikCubeSolved
    \RubikRotation{random,120}
    \SaveRubikState{exmpfig4.tex}
    \DrawRubikFlat
\end{tikzpicture}
\end{figure}
-------------------
```

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{exmpfig4.tex}`, and then input here using `\input{exmpfig4.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.

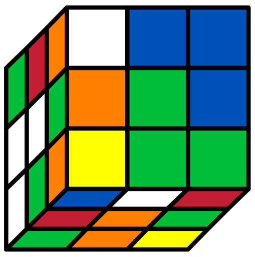


Figure 4: This shows a Rubik cube in exactly the same state as the one shown in Figure 3

```
-------------------
\begin{figure}[hbt]
\centering
\begin{tikzpicture}[scale=0.8]
   \input{exmpfig4.tex}
   \DrawRubikCubeLD
\end{tikzpicture}
\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**). Following each rotation we save the state to a file and then, finally, display each cube state in sequence, using a couple of useful macros (`\showcube`, and `\lift`), to 'show' the cubes and to 'lift' the rotation command slightly off the baseline. Note that embedding the TikZ picture environment in the `\showcube` command brings considerable flexibility.

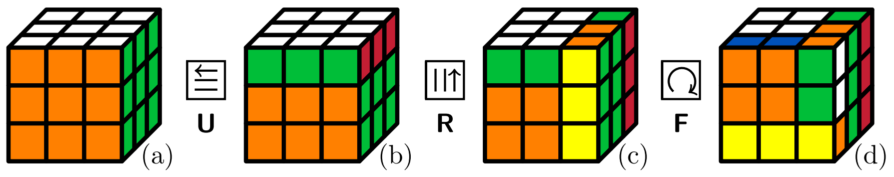


Figure 5: The rotations **U**, **R**, **F** on a solved cube.

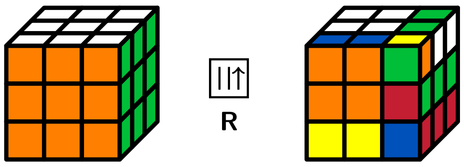
Note that even though the `\RubikCubeSolved` command (which allocates the colours to the 'solved' cube) is used prior to the TikZ picture environment, its effects are still available to the `\Draw...` command inside each of the picture environments.

```
-------------------
\newcommand{\showcube}[1]{%
   \begin{tikzpicture}[scale=0.5]
   \input{#1}
   \DrawRubikCubeRU
   \end{tikzpicture}
}
\newcommand{\lift}[1]{\raisebox{8mm}{#1}\hspace{2mm}}
%%
\RubikCubeSolved  \SaveRubikState{exmpfig5a.tex}
\RubikRotation{U} \SaveRubikState{exmpfig5b.tex}
\RubikRotation{R} \SaveRubikState{exmpfig5c.tex}
\RubikRotation{F} \SaveRubikState{exmpfig5d.tex}
%%
\begin{figure}[hbt]
   \centering
   \noindent     \showcube{exmpfig5a.tex}\hspace{-4mm}(a)
   \lift{\RubikU}\showcube{exmpfig5b.tex}\hspace{-4mm}(b)
   \lift{\RubikR}\showcube{exmpfig5c.tex}\hspace{-4mm}(c)
   \lift{\RubikF}\showcube{exmpfig5d.tex}\hspace{-4mm}(d)
   \ShowRubikErrors
\caption{The rotations \rrU, \rrR, \rrF\ on a solved cube.}
\end{figure}
------------------
```

Example 6

In this example we show how commands used inside a TikZ picture environment remain local to that picture environment.

In this case the first fig uses the `\RubikCubeSolved` inside the TikZ environment. However, if we then perform the rotation **R** $\boxed{||\uparrow}$ 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 state of the next earliest figure (ie Figure 5d), hence the strange form of the second figure below.



Since we are not here using any 'minipage' environments, we need to use the `\lift{}` command to raise the $\boxed{||\uparrow}$ hieroglyph off the baseline.

```
--------------------
\begin{tikzpicture}[scale=0.5]
   \RubikCubeSolved
   \DrawRubikCubeRU
\end{tikzpicture}
\hspace{5mm}\lift{\RubikR}\hspace{5mm}%
\RubikRotation{R}%
\begin{tikzpicture}[scale=0.5]
   \DrawRubikCubeRU
\end{tikzpicture}
--------------------
```

—— END ——