The **Rubik** User's Guide

Tom Davis
tomrdavis@earthlink.net
http://www.geometer.org
May 10, 2003

1 Introduction

The program **Rubik** allows you to manipulate a virtual 3x3x3 Rubik's cube. It runs on either a Mac (running OS X, version 10.2.4 or later) or a PC and provides the following features:

- Manipulate the cube (twist faces, slices, and the whole cube, and undo your twists).
- Jumble the cube.
- Reset the cube to solved.
- Solve a jumbled cube (whether it was jumbled by you or by the computer).
- Enter your own cube configuration (to find a solution for your own physical cube, for example).
- Define and apply macros. Macros are sets of twists grouped together and assigned a name so that the set of moves can be applied as a unit.
- Determine the order of a macro. The order of a macro is the number of times you need to repeat the macro from a solved cube to return to a solved cube.
- Run the program in a mesmerizing demo mode.

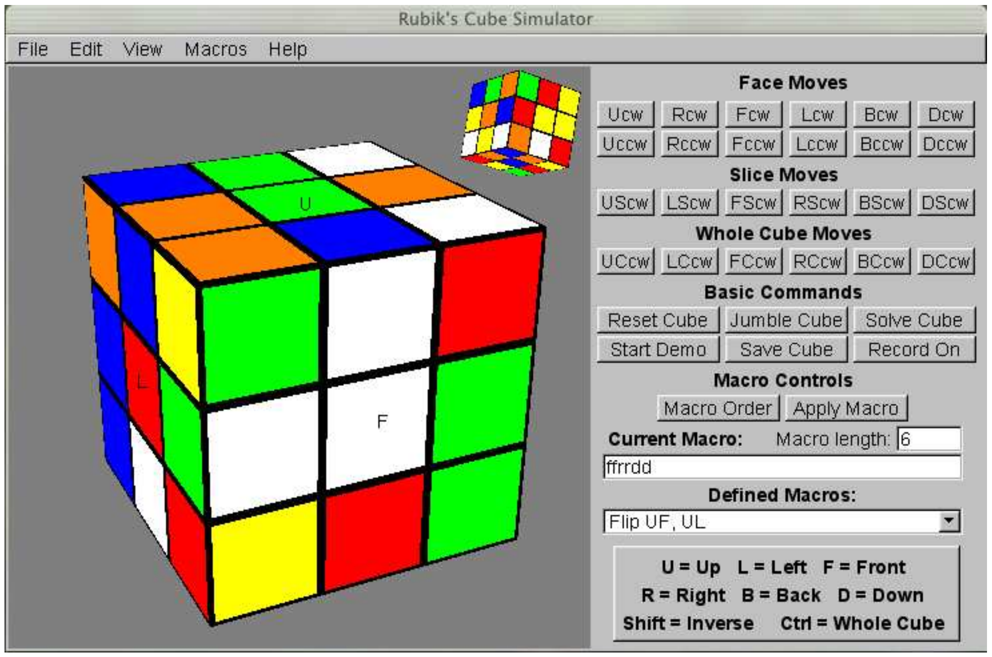
The **Rubik** program is based in part on the work of the FLTK project (http://www.fltk.org)—a darned good piece of software.

2 Basic Cube Manipulation

If the cube turns too quickly or too slowly, see the description of the $\boxed{\text{Speed}}$ button in the description of the commands in Section 12.

To run the program, just double-click the **Rubik** program icon. You should see a window something like that in figure 1 (except that the displayed cube will not be jumbled when the program starts).

Two cubes are visible—the large one can be manipulated and the smaller one is a view of the back of the same cube. Remember that it is truly a back view—the blue side is the right face and the orange side is the back face.



Current Macro: Macro length: 6

ffrrdd

Defined Macros:

Flip UF, UL

Figure 1: **Rubik** Window

To rotate a face 90 degrees clockwise (as you look at that face) click on the center cubie[1] on that face with the mouse. If you hold the **shift** key down while you click on a face, it rotates 90 degrees counter-clockwise. To obtain a half turn, simply click two times on the face's center cubie.

In general, if the **shift** key is down, the inverse operation is performed. For example, if a command rotates the entire cube right, that same command with the **shift** key down will rotate the entire cube to the left and so on. This is nice in that if you screw up a movement, you can almost always hold the **shift** key down and "repeat your mistake" to undo it.

If you hold down the **control** key while you click on the center cubie of a face, the entire cube turns 90 degrees clockwise. If you hold both the **control** and **shift** keys down when you click, the entire cube is rotated by 90 degrees counter-clockwise.

If you hold down the **alt** key while you click on the center cubie of a face, the center slice parallel to that face turns 90 degrees clockwise. If you hold both the **alt** and **shift** keys down when you click, that same center slice is rotated by 90 degrees counter-clockwise.

Finally if you hold down both the **control** and **alt** keys, the face opposite is turned 90 degrees from the point of view of the face you clicked on. For example, if you click on the front face with the **control** and **alt** keys down, the back face will rotate clockwise as it appears on the screen. Remember, however, that if you were looking directly at the back face, this would appear to be a counter-clockwise twist. As before, the addition of the **shift** key reverses the direction of the

[1]The name "cubie" applies to any of the smaller outer cubes that make up the entire $3 \times 3 \times 3$ cube.

turn.

To undo moves, use the $\boxed{\text{Undo}}$ entry in the $\boxed{\text{Edit}}$ pull-down menu or type the **control-Z** shortcut. You can issue undo commands repeatedly and each undoes one move. (If you just want to undo a single move, it's usually easier just to repeat whatever you did three more times (or repeat it once with the **shift** key down) since then you don't have to move the mouse to find the $\boxed{\text{Undo}}$ menu entry or reach for the keyboard to type **control-Z**.)

All of these operations can also be applied with buttons in the control area to the right of the window. As before, if the **shift** key is down, the movements are reversed.

Finally, there are shortcut keys for any of the basic face movements (not the slice or whole-cube movements, however). If you hold down the **alt** key and type "**f**", "**r**", "**d**", "**l**", "**b**" or "**u**" then the front, right, down, et cetera face is turned 90 degrees clockwise. If the **shift** key is down, then that face is turned counter-clockwise.

3 Cube Coordinates and Move Descriptions

Although at first glance, it seems like it might be best to describe elements of a cube by their colors, this is not a good idea for two reasons:

1. Different cubes with different manufacturers have different arrangements of the face colors.
2. If you've learned a sequence of moves that, for example, exchanges two adjacent corners, this same operation can be applied to *any* pair of adjacent corners. A color-based description would only apply to one pair.

If you hold a cube in front of you with one face toward the ceiling and looking directly at another face, there are six faces altogether that we will call "Up", "Down", "Left", "Right", "Back" and "Front". In the intial cube configuration the default face colors of **Rubik** has the white face "Up", the red face "Front" and the green face "Left". Although you cannot see them except in the rear view, the blue face is "Right", the orange face is "Back", and the yellow face is "Down".

If your cube is not arranged like this, you can reset the cube colors so that the program's cube will look like your physical cube. See Section 10.

These names of the faces are fairly standard, and they are abbreviated by their first letters which are luckily all different: B F U D L R. The other great thing about the names is that the letters in the order above almost spell "befuddler" of which the cube surely is one.

To help avoid errors, the Front, Left and Up faces are labeled with the letters "F", "L" and "U" on the computer display.

The buttons in the control area refer to these coordinates. For example, the button labeled $\boxed{\text{Dcw}}$ means to turn the Down face 90 degrees clockwise. That's clockwise when you are looking at the middle of the Down face. Similarly, $\boxed{\text{Bccw}}$ means turn the back face 90 degrees counter-clockwise (as you look at it—if you're viewing from the front, it appears to be a clockwise move[2]). The button marked $\boxed{\text{RScw}}$ means to look at the right face and turn the center slice 90

[2]If you are manipulating a physical cube, an easy way to keep clockwise and counter-clockwise straight is to remember that to get a clockwise move of a face, grasp it with the right hand and turn it in the direction pointed to by the right

degrees clockwise. Finally, the $\boxed{\text{UCcw}}$ means to look directly at the Up face and turn the entire cube 90 degrees clockwise. Remember that if the **shift** key is held down when any of these buttons are pressed, the movement is in the opposite direction. (If you are befuddled by this paragraph, just try clicking on the faces with various combinations of **shift**, **control** and **alt** and clicking on the buttons in the control area with and without the **shift** key and it will become clear.)

Note that there's no need for counter-clockwise slice or whole-cube moves; $\boxed{\text{BScw}}$ is the same as turning the Front slice counter-clockwise and similarly for whole-cube moves. You can also hold down the **shift** key to get the reverse movement.

The cube appears to be made of 27 smaller "cubies", of which you can see the 26 outer ones. (There really is no center cubie, but that doesn't matter.) There are three kinds of visible cubies: corner cubies with three colors (there are eight of these), edge cubies with two colors (there are twelve of these) and face cubies with one color (there are six of these).

A cubie can be identified by the three, two or one colors (or better, by the three, two or one faces. The UFR cubie is the corner cubie in the upper-front-right. The LD cubie is an edge cube on the left-down edge. The R cubie is the center cubie on the right face. See figure 2.

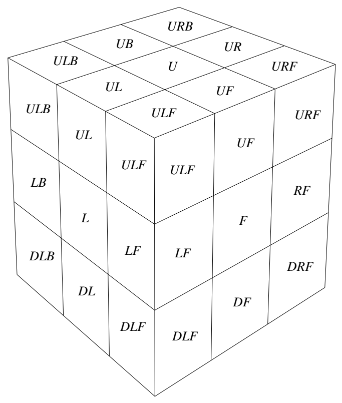


Figure 2: Cube Coordinates

With the default cube as displayed in the program, the LUF cube is the one with the green, white and red faces showing. (It's the only corner cubie all of whose faces are visible on the large cube.) The three edge cubies next to it are the white-red (UF), white-green (UL) and green-red (LF) cubies.

If you are just talking about a particular cubie whose orientation is unimportant, then the order of the letters does not matter: URF, UFR, RUF, RFU, FUR and FRU would refer to the same cubie. If you wish to describe a transformation exactly it can be done by paying careful attention to the order of the letters. For example, a transformation that flips the UL and UF cubies in place (in other words, the cubes stay in the same locations on the cube, but the colors are exchanged) might be written like this:

$$(UL, UF) \rightarrow (LU, FU) \text{ or } (UL)(UF)$$

where the notation indicates that the cubies stay in the same locations, but the faces are flipped. The second possibility is in "cycle notation" where each element in a pair of parentheses is assumed to move to the element to the right and the rightmost returns to the first element. Since there is only a single face, it's clear that the notation refers to a cubie that stays in place but changes orientation.

thumb.

In the same way, a transformation that rotates a pair of corner cubies in place might be written:

$$(UFL, UFR) \rightarrow (LUF, RUF) \text{ or } (UFL)(UFR).$$

Finally, a transformation that cycles three edge cubies RF to UF, UF to FL and FL to RF might be written:

$$(RF, UF, FL) \rightarrow (UF, FL, RF) \text{ or simply } (RF\ UL\ FL).$$

You can find this cycle form of the permutation that will take a solved cube to the one displayed on your screen by using the $\boxed{\text{Display Permutation}}$ command in the $\boxed{\text{File}}$ pull-down menu. The permutation is displayed in a window that pops up.

The permutation of the cube is displayed in two forms. The first indicates how the cubies move on the cube. This is the way most people think about cube permutations, but it cannot completely represent the situation. We can indicate the how a face is rotated or flipped if it is in its original location, or we can indicate how the individual cubies move into their slots, but not both. Here, if the cubies move, the permutation indicates the cycling of the cubie locations. If a cubie stays in one place, the permutation indicates how the faces of that cubie are rotated or flipped. For example, (RUF) indicates that the cubie is in and remains in the right-up-front corner, but that the right face moves to the up face, the up face to the front face, and the front face back to the right face. However, $(RF\ UF\ FL)$ simply tells us that those three cubies cycle in the manner shown, but it gives no indication of whether they are flipped or not.

To obtain a complete description of the permutation, we need to know where each of the 54 faces move. Well, actually only 48 of them, since we consider the center cubies on each face to stay in place. The second form of the permutation displayed indicates this. Each corner cubie (like RUF) has three faces. For that example, **Rubik** uses Ruf to indicate the face on the right, Urf for the up face, and Fur for the front face of that same cubie. Similarly for the edge cubies.

After each of the two cycle notations listed above is an indication of the cycle structure. In that structure, a term like $3(4)$ means there are 3 cycles of length 4. Cycles of length 1 are generally not listed since they indicate that an object stays in place, but in the case of the cubie permutation, 1-cycles indicate the number of cubes that stay in place but are flipped or rotated. For details you can always look at the cubie face cycle notation.

4 Solving the Cube with Macros

A "macro" is simply a series of twists considered to be a single operation. If, for example, you turn the front face twice by 90 degrees and then the left face twice by 90 degrees, that would be a four-move macro. This is not a particularly useful macro, but it will serve as a good example.

Try this out: Press the $\boxed{\text{Reset Cube}}$ button to initialize the cube, and then click twice on the front center cubie (the red one) and then twice on the left center cubie (the green one). The cube looks a bit jumbled, but it's still quite regular. (The shortcut for $\boxed{\text{Reset Cube}}$ is **control-R**.)

Now repeat the operation again (Front, Front, Left, Left) so that now you've done 8 total quarter-turns. The cube is still mixed up, but it's not that much worse. Repeat the four quarter-turns one more time (for twelve total turns) and the cube is almost solved. If you examine the cube's front and rear views, you should be able to see that there are only four cubies out of place. Now do the entire four-move macro three more times (that's twelve more total twists), and if you didn't make any mistakes, the cube returns to its "solved" configuration. Pretty neat, huh?

It's a little inconvient to do four clicks each time you want to apply this macro, so let's see how to do it more easily.

There is a macro language that describes the steps in a macro, and it's fairly straight-forward:

- B, F, U, D, L, R: $90^{c}irc$ clockwise move of the corresponding face.
- b, f, u, d, l, r: 90° counter-clockwise move of the corresponding face.
- *B, *F, *U, *D, *L, *R: 90° clockwise slice move.
- $>$B, $>$F, $>$U, $>$D, $>$L, $>$R: 90° whole-cube move.

To construct a macro, just cram together the appropriate letters in a string. Your "Front Front Left Left" macro is just:

FFLL

Let's see how to use it. First, click on $\boxed{\text{Reset Cube}}$ to initialize the cube.

Click on the empty $\boxed{\text{Current Macro}}$ input box (if it's not empty, simply select all the text in it and type the **delete** key). Next, type in "FFLL" (without the quotation marks, of course). If you now press the **return** key on your keyboard, the macro is applied, and you can watch the faces rotate.

Notice that just above the right end of the $\boxed{\text{Current Macro}}$ input box is an output area that lists the length in steps of the macro. This can be valuable if you want to find macros that accomplish various tasks in the minimum number of steps.

In fact, if you press the **return** key again, the macro is repeated. Thus, clicking on the **return** key six times will do exactly the same thing that your 24 individual moves did previously.

If you are in a hurry, click on the $\boxed{\text{Apply Macro}}$ button in the control area, and the macro is simply applied to the cubies and all you see is the final result—you don't need to wait for them to turn for you. Try that—with the same FFLL in the macro window (from a reset cube) click the $\boxed{\text{Apply Macro}}$ button 6 times to see how it works. The shortcut for $\boxed{\text{Apply Macro}}$ is **control-A**.

Now let's see how macros can be extremely useful for solving a cube. Reset the cube (with the $\boxed{\text{Reset Cube}}$ button) if it is not already, and click on the "Flip UF UL" entry in the menu button labeled $\boxed{\text{Defined Macros}}$.

Two things happen. First, whatever you had in the $\boxed{\text{Current Macro}}$ area is replaced by this 14-step macro:

FRBLUlUbrfluLu

In addition, the cube is modified to represent the result after these 14 moves have been applied. As you can see, this macro might be *very* useful for a solution. Any time you have two adjacent edge cubies that are in the correct spots, but flipped to the wrong orientation, you could place the cube as shown on the screen and apply those 14 moves with the result that your two edge cubies would be flipped in place.

Click on the “Flip UF UL” choice button again and the macro is applied a second time, returning the cube to its initial configuration.

Now click in the $\boxed{\text{Current Macro}}$ area where you previously typed in the FFLL and type the **return** key. This time the 14 steps are applied one at a time so you can see what happens to the cube as you step through the macro. Any time you want to watch a macro occur, click in that $\boxed{\text{Current Macro}}$ area and either enter your own macro definition or use one that's there and then press the **return** key to step through it. If you hold down the **shift** key when you press **return**, the inverse of the macro will be applied to the cube. In other words, if you start with a solved cube, then execute a macro, then hold down the **shift** key and press the **return** key again, the cube will be returned to a solved state.

The $\boxed{\text{Apply Macro}}$ button is also inverted if the **shift** key is down when it is pressed.

There are a few other macros stored in that list by default. To try others, click down in the $\boxed{\text{Defined Macros}}$ menu choice button, drag the mouse to be above the one you want, and release the mouse button. That macro will be applied as well as entered in the $\boxed{\text{Current Macro}}$ area.

To get a good idea of what each does, initialize the cube with the $\boxed{\text{Reset Cube}}$ button before you try each one. All are useful tools that might be used to solve a cube (real or virtual).

4.1 Defining Your Own Macros

If you have found a useful set of moves that you would like to be able to reuse, type it into the $\boxed{\text{Current Macro}}$ area, test it to be sure it does what you want, and then click on the $\boxed{\text{Define Macro}}$ entry in the $\boxed{\text{Macros}}$ pull-down menu. A text entry window will come up, allowing you to name your macro, and when you've named it, it will appear in the list of macros.

As soon as you define a new macro, it is written into the preferences file so that from then on, when you start **Rubik**, you will have those macros automatically defined for you.

To delete a macro from the list in $\boxed{\text{Defined Macros}}$, click on the $\boxed{\text{Delete Macro}}$ entry in the $\boxed{\text{Macros}}$ pull-down menu. Then type in the name of the macro (exactly as it is in the list, including spaces), and the macro will be deleted, not only from the current list, but also from the preferences file if you wish. Notice that the default name for the macro to delete is the last one you used, so the quickest way to delete a particular macro is to use it and then delete it. If you don't want to alter the cube by applying the macro, simply apply the macro and then apply it again with the **shift** key down. This will do and undo the macro action for no net effect.

You can delete any of the macros, and if the one you delete is one you defined, **Rubik** will ask you if you want to delete it from the preferences file as well. You can also delete the macros that appear in **Rubik** by default, but they will reappear the next time you run the program.

In addition, the command $\boxed{\text{Delete All Macros}}$ gets rid of all the macros in the choice button. This does not delete any of the macros you defined from the preferences file.

Remember that if you applied your FFLL macro three times, it did something fairly simple—it exchanged two sets of edge cubies and left everything else in place. This might actually be useful for cube solution, and so you could create a macro that does the twelve turns by typing in:

FFLLFFLLFFLL

and defining a macro. But there's a simpler way. Just type:

3(FFLL)

A number before a pair of parentheses says to repeat the stuff inside the parentheses that number of times. Again, here it's a bit of overkill to use it, but try typing 3(FFLL) into the $\boxed{\text{Current Macro}}$ area, test it, name it, and use it.

In fact, such grouping can be nested. Again, this is harder than doing it in the straightforward manner, but the following represents the same macro:

3(2(F)2(L))

Obviously, for more complex situations, this may save a lot of time.

Remember that you can use upper-case letters for counter-clockwise moves, and you can include slice and whole-cube moves in your macros. Here's a valid macro (not useful, but valid):

3(f>DRR*Uu3(RF)2(FU))rfr21(FRFuud)

If you type junk like that into the $\boxed{\text{Current Macro}}$ area and press the **return** key, it'll give you a "show". In this case, the macro above results in 177 individual moves. Do you see why?

We noticed that if you repeat the sequence FFLL six times from an initialized cube, the cube is returned to "solved". What is interesting is that *every* macro has similar behavior: if you repeat it enough times from a solved cube, the cube will return to solved. The minimum number of times required to return to the "solved" configuration is called the "order" of the macro. The order of FFLL is thus 6.

Rubik allows you to find the order of any macro. Type the macro into the $\boxed{\text{Current Macro}}$ input area and press the $\boxed{\text{Macro Order}}$ button. The result is displayed in a window. Try entering FFLL in the $\boxed{\text{Current Macro}}$ area, press $\boxed{\text{Macro Order}}$ and see that the result is 6.

If you enter just FL, the order is 105—probably larger than you thought. You can easily verify that 105 repetitions of the FL macro returns to "solved" by resetting the cube, typing 105FLl) into the $\boxed{\text{Current Macro}}$ area, and then pressing the $\boxed{\text{Apply Macro}}$ button. (It's a little tougher to verify that 105 is the *minimum* number of applications of the macro required to return to solved—you'll just have to trust **Rubik**.)

Being able to find the order of a macro is useful for solving cubes since if the order of a macro has divisors, going only half-way or a third of the way through a macro often has useful results. For example, if you initialize the cube and try the combination FLfl, it moves 7 total cubies—not too useful for solving a cube. But if you type it into the $\boxed{\text{Current Macro}}$ area and press the $\boxed{\text{Macro Order}}$ button, you'll see that the order is 6. If you now try the macro 3(FLfl) you'll see that it swaps two pairs of corner cubies, and that might be useful.

4.2 The Macro Gizmo

Sometimes you would like to have a set of macros all available at the touch of a button. **Rubik** can display a macro window where you can define up to 16 such macros. All the macros in the gizmo must be defined macros.

The gizmo holds up to 16 macros. You can enter more, but only the first 16 are displayed.

To enter a macro into the gizmo, use the $\boxed{\text{Add Macro to Gizmo}}$ entry in the $\boxed{\text{Macros}}$ pull-down window. Enter the name of the macro you would like to add in the window that pops up. The default name in that window is the macro currently visible in the $\boxed{\text{Defined Macros}}$ choice area. As soon as you've added a macro, the macro gizmo appears with the new macro name listed on a button. Pressing this button is the same as clicking on the name in the $\boxed{\text{Defined Macros}}$ area, including the feature that if the **shift** key is down, the inverse of that macro is applied.

To delete a macro from the gizmo, use $\boxed{\text{Delete Macro from Gizmo}}$ in the $\boxed{\text{Macros}}$ pull-down menu. This does not delete it from the set of defined macros—only from the gizmo.

If you have closed the macro gizmo window, you can re-open it with the $\boxed{\text{Show Macro Gizmo}}$ button in the $\boxed{\text{View}}$ pull-down menu. If there are no macros in the gizmo, it will not be displayed.

This gizmo is useful when you are running experiments with a restricted set of moves. For example, suppose you would like to see what positions can be achieved using only the three slice moves. Put each one in a macro and enter exactly those three into the gizmo. Then click only on the gizmo buttons. Hint: if you move the gizmo to sit on top of the control area of **Rubik**'s window, you won't accidentally use any of the normally-available buttons by accident.

To clear the macro gizmo, use the $\boxed{\text{Clear Macro Gizmo}}$ command in the $\boxed{\text{Macros}}$ pull-down menu.

If (and only if) the macro you add to the gizmo is one that is in your preferences file, a change will be made to your preferences file so that it will be loaded into the gizmo each time you start up **Rubik**. If you delete this macro from the macro gizmo, the preferences file will be changed so that it is not automatically loaded into the gizmo on startup. You can temporarily add macros to the gizmo from the default startup set or from macro files that you have loaded, but that information is lost the next time you start up **Rubik**. If you really want one of those macros in the gizmo permanently, define your own copy of it and put that in the gizmo. Remember that if the macro string in a file that you load begins with the character "$", then the macro is automatically loaded into the gizmo when the file is loaded.

4.3 Singmaster Notation

David Singmaster was one of the first people to publish a booklet on Rubik's cube, and he used a notation that is different from that used in **Rubik**. If you search on the web you will find many interesting macros described in this so-called Singmaster notation, so **Rubik** has the ability to interpret it.

Singmaster's notation is quite simple. It consists of a string of space-separated entries that look like this: U, U', U2, R, R', R2, et cetera. The Singmaster U is **Rubik**'s U, the U' is **Rubik**'s u and finally, U2 is **Rubik**'s UU. Similarly for the other 5 faces/letters.

To enter a macro in Singmaster notation, simply surround it with brackets when you type or paste it into the $\boxed{\text{Current Macro}}$ area, as follows:

[U D R2 L' B F' U2]

As you type it into the $\boxed{\text{Current Macro}}$ input area (or paste it in with **control-V** key) nothing happens until you type **return** or click on the $\boxed{\text{Apply Macro}}$ button. At that point the Singmaster notation is translated to the **Rubik** notation and the contents of the input area are changed to be in **Rubik** format, at which point you can do anything you like with them.

To copy text from the $\boxed{\text{Current Macro}}$ area, select it and type **control-C**. To cut it, type **control-X**.

5 Recording Macros

If you press the $\boxed{\text{Record On}}$ button in the control area, **Rubik** will record your cube movements in the $\boxed{\text{Current Macro}}$ input area. It uses exactly the same syntax to record the moves, and once the text is in that input area, it can be used like any other macro.

Doing almost anything other than making movements will get you out of this recording mode (but will leave the macro in the $\boxed{\text{Current Macro}}$ input area). If you want to force **Rubik** out of recording mode, click on the same button (whose name has changed to $\boxed{\text{Record Off}}$).

The $\boxed{\text{Undo}}$ command (in the $\boxed{\text{Edit}}$ pull-down menu) will erase the last entry in the Macro window as well as undoing the operation.

6 Macros from a File

If you would like to use different sets of macros for different runs of the **Rubik** program (perhaps you are writing a tutorial that uses the program, for example), you can do so. Create a text file with pairs of lines that contain first the name, then the macro itself. For example, such a file might look like this:

```
Three face twist
rld
Junk-filled macro
fuD>R*Duud*U*R*U
Striped Cube
[ D′ F D′ L B D2 F2 U R B′ U R2 F D′ R F U2 ]
```

This example would define a macro called "Three face twist" to be RLD and another macro called "Junk-filled macro" to be fuD>R*Duud*U*R*U. Finally, a third macro called "Striped Cube" is defined in Singmaster notation. Suppose this file is called `rubikmacs.rub`.

If the macro string begins with the character "$", that macro will also be added to the macro gizmo. See Section 4.2

To put the macros from a file into the $\boxed{\text{Defined Macros}}$ area, use the $\boxed{\text{Load Macro File}}$ entry in the $\boxed{\text{File}}$ pull-down menu. A file browser will appear that you can use to navigate to the proper directory and select the file `rubikmacs.rub`. The macros above will be added to your list of defined macros for the current session only. These macros are not stored in the preferences file, so they will disappear the next time you run **Rubik** and will have to be reloaded again if you want to use them again.

Currently **Rubik** does no checking to make sure that the names of macros are distinct. It's up to you to keep them organized. If you accidentally get a duplicate name, you can always delete it, although it's not clear which one will be deleted.

Included with the release is a file called `patterns.rub` that contains a number of macros that produce pretty patterns if they are applied to an initialized cube. Give some of them a try.

If you've got some macros that you developed within **Rubik** and would like to save them, the command $\boxed{\text{Dump Macro File}}$ in the $\boxed{\text{File}}$ pull-down menu allows you to specify a file name and *all* of the currently-defined macros are dumped there. You'll have to sort out the mess with a text editor, assuming you only want a few of them.

7 Demo Mode

Make sure that the $\boxed{\text{Current Macro}}$ area is empty and press the $\boxed{\text{Start Demo}}$ button. You'll get an interesting display. To stop the demo, click on the same button (whose name has been changed to $\boxed{\text{Stop Demo}}$). If you find the "orbiting" small rear-view of the cube annoying, you can toggle it on and off with the $\boxed{\text{Show Rear Cube}}$ entry in the $\boxed{\text{View}}$ pull-down menu.

If the $\boxed{\text{Current Macro}}$ area is empty, demo mode creates a random macro and repeats it endlessly, so it's guaranteed to come back to "solved" again and again. The macro it picks will return to solved in a reasonable number of turns (fewer than 300).

If the $\boxed{\text{Current Macro}}$ area is not empty when you press the $\boxed{\text{Start Demo}}$ button, the program initializes the cube and endlessly applies the macro entered there. Thus if you find a macro that goes though pleasing states, you can probably make a nicer demo than what the program is likely to come up with by itself (but sometimes **Rubik** will surprise you). You might try demo mode with some of the macros in the file `patternmacs.rub`.

If **Rubik** makes up a macro for use in demo mode, it puts it in the $\boxed{\text{Current Macro}}$ display area so in case it stumbles across a good one, you can save it.

8 Jumbling and finding Rubik's Solution

If you want to practice solving jumbled cubes, simply press the $\boxed{\text{Jumble}}$ button and you'll get a completely jumbled cube that you can try to solve with a combination of edge moves and macros. (If you pry apart your cube with a screwdriver, mix the pieces, and assemble it with your eyes closed, you have one chance in twelve of obtaining a solvable cube.)

Of course it's pretty easy for you to mix up the virtual cube hopelessly yourself with a few random clicks on the buttons in the control area.

If you want the program to solve a cube for you, Simply click on the $\boxed{\text{Solve Cube}}$ button. It may take a while. The first time you do this, there is also an initialization that occurs that also takes time. (On my dual processor 1.25 GHz Power Mac, it takes perhaps fifteen seconds to find the solution for an “average” cube (including initialization). On my old 400 MHz PC, it takes about 60 seconds.)

The solution appears in the Current Macro area. You can either make those moves yourself or click in the Current Macro area and press the **return** key to watch the program solve the cube. Don’t simply click on the Apply Macro button—that would just be the equivalent of pressing the Reset Cube button, right?

By the way, you can convince yourself that **Rubik** really is solving the cube and not just watching the turns you make as you jumble it and reversing those. Try starting with an initialized cube, make about 50 moves, and then click on Solve Cube. It will certainly find a solution that has fewer than 50 quarter-twists (**Rubik**’s solution is typically between 25 and 30 moves).

When you press the Solve Cube button, **Rubik** spends a “reasonable” amount of time looking for a good solution. If you’d like to have it spend more time and make it look for a better solution, there are two entries in the Edit pull-down menu: Long Solve and Very Long Solve. If these are toggled on, **Rubik** will spend more time and *much* more time searching for a solution. Even if **Rubik** is in one of these modes, a solution may not take longer—if **Rubik** is convinced that it has the best solution, the search can end early. You can convince yourself that this is true by making, say, three quarter-turns on a solved cube and then clicking on the Solve Cube button in Very Long Solve mode.

9 Solving a Physical Cube

If your plastic cube is jumbled and you’re too stupid to solve it yourself and too lazy to take out the screwdriver to break it apart and then reassemble it, here’s a way to solve it without wracking your brain too much.

Enter your cube’s configuration into the program and use the Solve Cube facility described above. When you see the macro in the window, manually apply that to your own cube by physically turning the sides, and there you are! Be sure that your cube is in the same orientation as the cube on the screen before you begin to apply the moves or you’ll wind up with a cube that’s simply jumbled in a different way.

To make sure you are following the macro exactly on your physical cube, especially when you are a beginner, you can use the **left-arrow** and **right-arrow** keys to single-step forward and backward through the macro. Thus, the safest way to solve your cube is to make one or two physical moves, then single-step the macro the same number of steps and compare the results. If they are different, you are at most a couple of moves away from the path to the correct solution. Remember that you can back up on the screen as well with the **left-arrow** key.

You can find the number of steps in the solution by looking at the “Macro length” output area above the Current Macro input area. If the macro doesn’t make sense (or is incomplete because you haven’t finished typing it in), then “*****” will be displayed as its length.

Remember, however, that one screw-up in the middle will totally trash everything and you’ll have to start again.

All you need to know now is how to enter your cube’s color configuration into the program.

For now, I will assume your cube has the same colors as those in the default cube (which, surprisingly, are the same as those on my personal physical cube).

But here's how it works with my cube (which is pretty standard, I think).

My cube's colors are Orange, Red, Green, Blue, White and Yellow. The first letters are all distinct, so I'll call them O, R, G, B, W and Y.

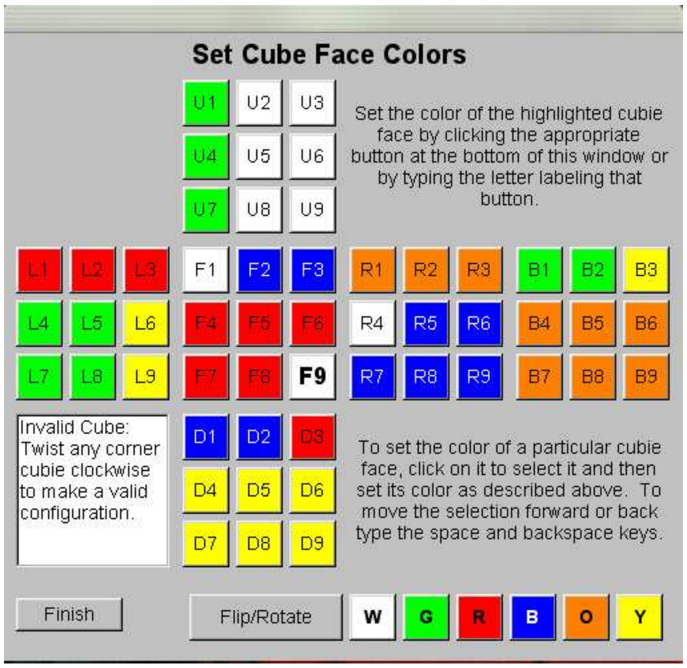
If the red center cubie is facing me and the white center cubie is up, then "Back" is orange, "Right" is blue, "Left" is green, and "Down" is yellow. If the cube were made of just a cardboard shell, and painted as in the "solved" configuration, you could cut it along various of its edges and unwrap it to look like this:

$$\begin{array}{cccc} & \boxed{\text{W}} & & \\ \boxed{\text{G}} & \boxed{\text{R}} & \boxed{\text{B}} & \boxed{\text{O}} \\ & \boxed{\text{Y}} & & \end{array}$$

(The figure above represents a configuration where the white face is on top, the red is facing you, the yellow is facing down, the green face is to the left, et cetera, just as in the default configuration on **Rubik**'s display.)

Now, Imagine that you again cut the cube as above, but this time draw out each face with the nine colors that make it up. When you do that, each of the letters in the pattern above would really represent 9 colors.

To enter your cube, click on the $\boxed{\text{Input Cube}}$ button and a window like that shown in figure 3 will appear.


Set Cube Face Colors

Set the color of the highlighted cubie face by clicking the appropriate button at the bottom of this window or by typing the letter labeling that button.

Invalid Cube: Twist any corner cubie clockwise to make a valid configuration.

To set the color of a particular cubie face, click on it to select it and then set its color as described above. To move the selection forward or back type the space and backspace keys.

U1 U2 U3
U4 U5 U6
U7 U8 U9
L1 L2 L3 F1 F2 F3 R1 R2 R3 B1 B2 B3
L4 L5 L6 F4 F5 F6 R4 R5 R6 B4 B5 B6
L7 L8 L9 F7 F8 F9 R7 R8 R9 B7 B8 B9
D1 D2 D3
D4 D5 D6
D7 D8 D9
Finish
Flip/Rotate
W G R B O Y

Figure 3: Set Cube Gizmo

One of the entries is highlighted (when you begin, it's the one in the upper-left corner of the "Up" face). At the bottom of the window is a set of color buttons. If you click on one of those, the color of the selected cubie face changes to be that color and the selected cubie face advances one. If you just want to change a few colors, you can click on a cubie face to make it the selected one, and continue from there. In addition, you can type the **space** and **backspace** keys to move the selected cubie forward or back without changing the colors. When you are satisfied with the cube coloring, click on the $\boxed{\text{Finish}}$ button.

Make sure especially not to screw up the back—when it's "unwrapped" it goes the "other way". If you have any doubts about how this works, the cube displayed in figure 1 is the same as the one displayed unwrapped in figure 3.

When you finish setting colors, **Rubik** does a "sanity check" on the cube colors to make sure you have the right number of cubies colored in a way that the cube can be solved. If you do not, **Rubik** displays an error dialog and returns you to the input gizmo. If you just can't figure out how to fix your cube, press the $\boxed{\text{Reset}}$ button in the gizmo and the cube will revert to a solved state.

If the cube in the input gizmo is invalid for some reason, that reason will be presented in the text output area in the lower-left part of the window. **Rubik** will list in that window one thing that is wrong. There may be many problems, but as you fix each, the next appears in that window. When it is completely valid, the text in that window will be "Valid Cube".

The gizmo also contains a $\boxed{\text{Flip/Rotate}}$ button. If you press this, the currently-selected cubie will flip (if it's an edge) or rotate (if it's a corner). Rotation is always in the same direction, so if you want to rotate in the opposite direction, just click on the button twice.

10 Setting Cube Colors

If your cube's colors don't match the default colors, you can change the default color settings.

Here are the colors currently recognized by **Rubik**, and their one-letter abbreviations (which can be in lower-case as well):

R: red	W: white	B: blue	Y: yellow	G: green
O: orange	S: silver	K: black	V: violet	P: pink
D: gold	M: magenta	X: gray	C: cyan	

To specify a coloring, list the letters corresponding to your cube's colors in the following order:

Up Left Front Right Back Down

The defaut colors are thus: WGRBOY. If your cube is red on top, white on the bottom, black in front, pink in the rear, magenta on the left and gold on the right, your color pattern would be: RMKDPX. These are the letters and colors that will be displayed on the screen and on the gizmo that lets you enter cube configurations.

To change these default colors, press the $\boxed{\text{Set Cube Colors}}$ entry in the $\boxed{\text{View}}$ pull-down menu. Then enter your six-character string in the input window that appears. From then on, (until you set the colors again in the same way) **Rubik** will use that color scheme. When you set the colors, the result is also written into the preferences file.

11 Printing the Current Cube

In the $\boxed{\text{File}}$ pull-down menu is an entry: $\boxed{\text{Print EPS File}}$. This will cause **Rubik** to produce an encapsulated PostScript file that displays the current front face of the cube. You can use this in any way you want.

12 Button Reference

What follows is a short description of the actions of each of the command buttons in the control area and menu entries in the pull-down menu. One hopes that the actions of the face-turning buttons are obvious. The buttons are listed in alphabetical order.

$\boxed{\text{Add Macro to Gizmo}}$	adds a previously defined macro to the macro gizmo.
$\boxed{\text{Apply Macro}}$	applies the contents of the $\boxed{\text{Current Macro}}$ input area to the cube instantly, without letting you see the individual moves.
$\boxed{\text{Clear Macro Gizmo}}$	empties the macro gizmo.
$\boxed{\text{Define Macro}}$	lets you assign a name to the contents of the $\boxed{\text{Current Macro}}$ area and the (name, macro) pair is placed in the $\boxed{\text{Defined Macros}}$ menu button.
$\boxed{\text{Delete All Macros}}$	deletes all the macros in the $\boxed{\text{Defined Macros}}$ list. It does not delete any macros from your preferences file.
$\boxed{\text{Delete Macro}}$	deletes a macro in the $\boxed{\text{Defined Macros}}$ list. If that macro is in your preferences file, you will be asked if you want to delete it from there as well.
$\boxed{\text{Delete Macro from Gizmo}}$	deletes a macro from the macro gizmo. This has no effect on the currently defined macros.
$\boxed{\text{Display Permutation}}$	displays in cycle format the permutation required to bring a solved cube to the current configuration.
$\boxed{\text{Dump Macro File}}$	writes the macro definitions in a form that **Rubik** can read into a file whose name is specified by the user.
$\boxed{\text{Help}}$	is currently a pitiful place-holder for something better in a future release.
$\boxed{\text{Input Cube}}$	allows you to enter your cube's colors into the program for test manipulation or solution.
$\boxed{\text{Jumble Cube}}$	creates a solvable jumbled cube.
$\boxed{\text{Load Macro File}}$	loads a set of macros from a file.
$\boxed{\text{Long Solve}}$	increases the amount of time that **Rubik** will spend looking for a good solution. This button can be toggled on and off.
$\boxed{\text{Macro Order}}$	determines the order of the macro in the $\boxed{\text{Current Macro}}$ area.

$\boxed{\text{Print EPS File}}$	produces and encapsulated PostScript file of the front face of the current cube configuration.
$\boxed{\text{Record On}}$	starts recording movements in the $\boxed{\text{Current Macro}}$ input area. When in recording mode, this key changes to be $\boxed{\text{Record Off}}$.
$\boxed{\text{Reset Cube}}$	initializes the cube to "solved" and empties the undo buffer.
$\boxed{\text{Set Cube Speed}}$	sets the speed at which the faces turn. Smaller numbers represent faster turning. The number you type in is the number of steps during a turn. If your graphics is pretty good (as of 2003) a speed of about 80 is good. On ancient machines or bad graphics, a 10 or even 5 might be the way to go. When you set the speed, the new value is stored in the preferences file. Set the speed to whatever pleases you.
$\boxed{\text{Set Cube Colors}}$	sets the default colors on the faces of a solved cube.
$\boxed{\text{Show Macro Gizmo}}$	displays the macro gizmo window.
$\boxed{\text{Show Rear Cube}}$	toggles on and off the rear cube view.
$\boxed{\text{Solve Cube}}$	solves the current cube configuration. A macro representing the solution is then entered into the $\boxed{\text{Current Macro}}$ area. You can type **return** to watch it solve automatically or you can single-step it with the **right-arrow** key, which is probably the best approach if you're using **Rubik** to solve a physical cube.
$\boxed{\text{Start Demo}}$	enters demo mode. On entry, the button changes to $\boxed{\text{Stop Demo}}$.
$\boxed{\text{Quit}}$	quits the program.
$\boxed{\text{Undo}}$	undoes one quarter-twist.
$\boxed{\text{Very Long Solve}}$	dramatically increases the amount of time that **Rubik** will spend looking for a good solution. This button can be toggled on and off.