Introduction

Cube Explorer is not one of the many programs which just simulate the Cube, or solve scrambled Cubes with lengthy maneuvers. Cube Explorer implements a sophisticated and very powerful algorithm. Confronted with scrambled Cubes of your choice, it produces some of the best, shortest solving maneuvers ever seen. This algorithm quickly provides a solving sequence of 19 moves on average, usually only one or two moves more than the perfect solution. Occasionally, you will observe Cube Explorer achieving perfection itself: producing a solution provably impossible to surpass.

Cube Explorer also assists you in finding Pretty Patterns (and provides very short generating maneuvers to carry them out). Cube Explorer includes a unique algorithm to search the universe of Cubes. Given your instructions to guide its search, it produces a list of the best, most elegantly patterned Cubes possible.

Basic Definitions

The Cube consist of 8 corner cubies, 12 edge cubies and 6 center cubies. The edges can be flipped, and the corners can be twisted. That means the orientation of these cubies can change in space.

The 6 different faces are called U(p), D(own), R(ight), L(eft), F(ront) and B(ack). While U denotes an Up Face quarter turn of 90 degrees clockwise, U2 denotes a 180 degrees turn and U' denotes a quarter turn of 90 degrees counter-clockwise. A sequence like U D R' D2 of Cube moves is called a maneuver. The number of moves is called the maneuver length. A maneuver can be a solver or a generator. While a solver is a maneuver which restores the original state of the Cube when applied to a scrambled cube, a generator is just the inverse of a solver - it is applied to a clean cube and the result is a scrambled cube. This makes sense if you want to play around with Pretty Patterns.

Cube Solving Algorithms

Commonly an algorithm to solve the cube works through a series of stages, each stage fixing the location or orientation of some corners or edges. As you move along the lines prescribed by the algorithm, you can see each of the faces taking a single color. Though this is appropriate for restoring the Cube by hand, this kind of algorithm is not optimal with respect to the total length of the maneuvers, because it disregards the mathematical group structure of the Cube. This kind of algorithm usually requires more than 70 moves.

The first who developed an algorithm which took into account the mathematical structure of the Cube was the mathematician Morwen Thistlethwaite in 1980. He worked through a series of four subgroups of the cube. The information about how to do this was held in four tables, and a computer was the appropriate tool to use these tables. To restore the Cube took at most 52 moves.

Hans Kloosterman improved this to a maximum length of 42 moves in 1990 by a different organization of the tables.

I developed the Two-Phase Algorithm in 1992 for a machine with 1 MB of RAM. The current implementation is more powerful and takes into account the larger amount of RAM available today. An exact upper limit for the maneuver length cannot be given because each cube situation is dealt with individually and it is impossible to examine all situations. But on the average you will need less than 19 moves.

Richard E. Korf applied general ideas from the theory of heuristic search algorithms in artificial intelligence to the special case of the cube in May 1997 and found a shortest maneuver sequences for 10 random generated cubes with his program. The program ran on a Sun Ultra-Sparc Model 1 workstation and needed about 82MB of RAM. To find a solution for the random cubes it took an average time of about 5 days per cube, but there were situations where it will take months or years to find a solution. The algorithm he used turned out to be very similar to the one used in each phase of the Two-Phase Algorithm.

In July 1997, taking advantage of the symmetries of the cube, Michael Reid was able to reduce the size of the pattern databases by an algorithm similar to the last two above. He ran his program on a machine configured with 128MB of RAM and a 200 MHz Pentium Pro Processor. The program found a shortest maneuver sequences about twenty times as fast as Korf's program.

The Two-Phase Algorithm

The following description is intended to give you a basic idea of how the algorithm works. It is quite technical and might be difficult to understand. The following information is not necessary for you to use the program. So omit it, if you want.

If you turn the faces of a solved cube and do **not** use the moves R, R', L, L', F, F', B and B' you will only generate a subset of all possible cubes. This subset is denoted by G1 = <U,D,R2,L2,F2,B2>. In this subset, the orientations of the corners and edges cannot be changed. That means, the orientation of an edge or corner at a certain location is always the same. The four edges in the UD-slice (between the U-face and D-face) always stay in that slice.

In phase 1, the algorithm looks for maneuvers, which will transform a scrambled cube to a state in G1. This means, the orientations of corners and edges have to be restored and the edges of the UD-slice have to be transferred into that slice. The orientation of corners and edges and the location of the four UD-slice edges are mapped to natural numbers and a state of phase 1 is represented by a triple (x,y,z) In this abstract space, a move just transforms a triple into another triple (x',y',z'). All cubes of G1 have the same triple (x0,y0,z0) and this is the goal state of phase 1.

To find this goal state we use a search algorithm which, in terms of the current research, is called iterative deepening A* with a lowerbound heuristic function (IDA*). In the case of the Cube, this means that we iterate through all maneuvers of increasing length. The heuristic function h1(x,y,z) estimates for each cube state (x,y,z) the number of moves that are necessary to reach the goal state. It is essential for the algorithm that the h1 function never overestimates this number. The heuristics h1 allows pruning while generating the maneuvers, which is essential if you do not want to wait a very, very long time before the goal state is reached. The heuristic function is a memory based lookup table and allows pruning up to 9 moves in advance.

In phase 2 we restore the cube in the subgroup G1, using only moves of this subgroup. Now we have to restore the permutation of the 8 corners, the permutation of the 8 edges of the U-face and D-face and the permutation of the 4 UD-slice edges. We map the permutations to natural numbers and a state of G1 is described by a triple (a,b,c) analogous to phase 1. The heuristic function h2(a,b,c) allows pruning up to 14 moves in advance and is also memory based.

The algorithm does not stop when a first solution is found but continues to find shorter solution by using suboptimal solutions of phase 1 as well. For example, if the first solution has 10 moves in phase 1 and 12 moves in phase 2, the second solution has 11 moves in phase 1 and 5 moves in phase 2. The length of the phase 1 maneuvers increase and the length of the phase 2 maneuvers decrease. If the phase 2 length reaches zero, the last solution found was an optimal solution and the algorithm stops. But usually you will stop the algorithm before you definitely know that the last solution was optimal because you will not have the time to wait for that.

Understanding the Concept of Cube Explorer

Cube Explorer consists of three windows:

1. The Facelet Editor, which you use to edit the colors of a cube and solve it.
2. The Pattern Editor, which helps you to create Pretty Patterns.
1. The Main Window, where the output of the other two windows is displayed, waiting to be manipulated by you. You can left click on a cube to select/deselect it and right click on a cube to rotate, reflect or remove it or to transfer it to the Facelet Editor. Then when you choose **Generate selected Cubes** or **Solve selected Cubes** from the **Main Window** menu, Cube Explorer will find generators or solvers for the cubes you selected. In this context we also say that we generate or solve the selected cubes.

The main purpose of Cube Explorer is to explore Pretty Patterns and find short generators for them. For this reason the maneuvers displayed in the Main Window are usually generators. Starting with a cube in the solved position, a generator is a maneuver which will transform it into the position displayed in the Main Window. Conversely, starting with a cube in the position displayed in the Main Window, a solver is a maneuver which will transform it into the solved position. Solvers are marked as such in the Main Window and are produced when you choose "solve" rather than "generate" in the menu.

Installing Cube Explorer

Cube Explorer only needs the files CUBE.EXE, CUBE.HLP and CUBE.CNT to run. You should copy them into a folder. On the first run, 15 tables are computed and stored in your Cube Explorer folder. Depending on your hardware, this can take 3 to 10 minutes. The next time you run Cube Explorer, the tables are directly loaded from your hard disk into the memory, which is considerably faster. The tables require about 6 MB of memory. Cube Explorer has a very powerful algorithm implemented to solve the Cube, and these tables are an essential part of the algorithm. You should not try to run Cube Explorer with less than 16 MB of RAM. Good system performance is essential for the search algorithm. The tables must be held in RAM - swapping of virtual memory will slow down the performance dramatically. So if you start a search and the hard disk keeps on running, you should close all other applications and restart.

Cube Explorer will not alter or create files outside the folder it is in. So should you ever decide to uninstall Cube Explorer - just delete this folder.

Solving the Cube

One of the three windows you see when running Cube Explorer is called Facelet Editor.

The square titled **Selected Color** shows the current color you are using, which is always one of the colors of the six center cubies. If you click on a center cubie, the selected color will be changed to the color of this cubie.

You can change the color arrangement of the cube with **Facelet Editor|Customize selected Color**. A Color Dialog appears and you can alter the selected color. This feature is important, if your cube has a different color arrangement than the default cube in the Facelet Editor.

If you right click when the mouse cursor is over the Facelet Editor window, a pop-up menu appears, so it is not necessary to access the menu bar of the Main Window.

Solve Cube copies the cube from the Facelet Editor to the Main Window and then computes a solver and displays the maneuver, and its length, beside the cube. As the search continues, maneuvers with decreasing length are displayed. Before adding the cube to the Main Window, there is a check if an isomorphic cube cube already exists in the Main Window. You will get a message in this case.

Note that you can change the search time in the **Options** menu**.**

Add Cube to Main Window just copies the cube to the main window. Then you have the possibility to solve or generate this cube later from the Main Window menu.

Generate Random Cube is useful if you want to convince yourself of the performance of the algorithm. One cube is picked out of the 43,252,003,274,489,856,000 possible cubes at random.

Searching for Pretty Patterns

One of the three windows you see when running Cube Explorer is called Pattern Editor.

You choose a color by clicking on one of the rectangles on the right side of the Pattern Editor window. Note that it is unimportant which colors you use in the Pattern Editor, but only which colors are the same and which are different on the nine facelets of one face.

If you right click while the mouse cursor is over the Pattern Editor window, a pop-up menu appears, from which you can start the pattern search. All possible cubes will be computed and added to the Main Window. If there are isomorphic cubes solutions, only one of them will be added to the Main Window.

If you color only one of the four squares shown in the Pattern Editor, all cubes generated have this color pattern on *each of the six faces*.

If you color two or more of the four squares, each face of a generated cube matches one of the color patterns. The easiest way to understand the principles will be to play a bit with the Pattern Editor and to watch the output in the Main Window.

A pattern is called continuous if two neighboring faces mach along the edge. That is to say, two adjacent edge cubies and corner cubies have the same color on the first face if and only if they have the same color on the second face. Continuous patterns are good candidates for exceptionally beautiful patterns.

Searching for Generators

If you choose **Generate selected Cubes** from the Cubes menu, generators for all selected cubes in the Main Window will be computed. You select or deselect cubes by left clicking on them. If you right click on a cube, a pop-up menu will appear. You can choose to remove, rotate or reflect the cube, or to transfer it to the Facelet Editor for further editing. If you rotate or reflect a cube, you might not get the result you expected. The reason is that the cube not only is rotated or reflected, but also recolored, so that the colors of the center cubies always stay the same.

An important parameter is the Timeout for each search, which you will find under **Options| Parameters** in the main menu. The average maneuver length will decrease if you increase the Timeout. With the Multiple Direction Search Box checked, the search is done in three different positions of the Cube. This is a consequence of the asymmetry in phase 2 of the Two-Phase Algorithm. You should check this box if you are interested in very short maneuvers.

If there is a ‘ * ‘ at the end of the computed generator, the generator is proven to be optimal. Usually this will happen only with rather short maneuvers, not exceeding 14 moves and depending on the Timeout you choose.

The decimal point displayed in a generator or solver just divides the output of the two parts of the Two-Phase Algorithm.

Using Cube Data Files

You can save the generated maneuvers in the Main Window with **File|Save Generators**. The saved file is an ordinary text file which can be edited with any text editor. You can add a name for each generated maneuver in angle brackets, for example D F2 D' . R B2 R' D F2 D' R B2 R' (12) <Meson>. When you load the file with **File|Load Generators**, the name will be displayed in the Main Window.

Also, you can use the **File|Load Generators** feature to import generators from other sources. All moves must be separated by spaces. Maneuvers must be on separate lines. Then you can use **Main-Window|Generate selected Cubes** and try to improve the generators.

When you import generators, instead of using R', you also can use R3 or R- . You also can use the common abbreviations Ra instead of R L and Rs instead of R L'. Note, that you must write Ra' and not R'a.

Frequently Asked Questions

Q: When I compute some Pretty Patterns with **Pattern Editor|Search Patterns** and save them using the **File** menu, loading them again does not seem to work!

A: You actually do not save the patterns but the generators for them. So unless you do not use **Main-Window|Generate selected Cubes**, the results will not be saved.

Q: The search for Pretty Patterns added many cubes to the Main Window. How do I remove the cubes I do not like?

A: Right click on the cube you want to delete and use **Remove Cube** from the pop-up menu. To reduce the output of the patterns search, you could also check the **continuous** box in the Pattern Editor, which usually leads to the most beautiful patterns.

How to Support Further Development of Cube Explorer

Cube Explorer is freeware. It was fun to develop the algorithm and to see how well it performs, but to extend the idea to a user-friendly program was work. If you like Cube Explorer and you install it on your system you can show your appreciation for this work in any form you like. Just ask yourself: What the program is worth to me?. You can send a nice postcard from where you live, an interesting puzzle or book, money or whatever you like to my physical mail address:

Herbert Kociemba
Bismarckstr. 31
D-64293 Darmstadt
Germany

Any suggestions for improvements should be mailed to

kociemba@hrz1.hrz.tu-darmstadt.de

The latest version of Cube Explorer will be available at

http://home.t-online.de/home/kociemba/cube.htm

Special thanks to Lawrence Leinweber, whose suggestions for improvements in the user interface have influenced version 1.5 of Cube Explorer.

If you have generated, for example, the first three moves of a 10 move maneuver and the heuristic function tells you that it takes at least 8 moves from here to (x0,y0,z0), you do not need to generate the fourth move any more.

There are 3^7 different orientations of the corners, so 0 <= x < 3^7. We have 2^11 different orientations of the edges, so 0 <= y <2^11. To place the four UD-slice edges (disregarding their order within the slice) gives 12*11*10*9/4! possibilities, so 0 <= z < 495 .

A move in G1 does not change the orientation of the edges or corners, so x and y do not change. A move in G1 changes the location of the UD-slice edges in that slice, but z does not take into account the permutation of the edges, so z also does not change.

There are many maneuvers to solve a given cube. But no maneuver will have a shorter length than the computed solution.

For those, who want to know it even more exactly:
$h1(x,y,z) = \max\{h1(x,y), h1(x,z), h1(y,z)\}$,
where for example $h1(x,y)$ is the minimum number of moves to transform a state (x,y,z) to $(x0,y0,z')$ for any z' (this means, the z coordinate is ignored).

There are 8! different permutations of the corners, so 0 <= a <8!, we have 8! different permutations of the U and D edges, so 0 <= b < 8! and 4! different permutations of the UD-slice edges, so 0 <= c < 4!. We define h2(a,b,c) = max{h2(a,c),h2(b,c)}, where for example h2(a,c) defines the minimum number of moves to transform a state (a,b,c) to (a0,b',c0) for any b' (this means the b coordinate is ignored). (a0,b0,c0) denotes the solved cube here.
We cannot use h2(a,b), because the corresponding table would hold 8!*8! entries, which exceeds the available memory.

Two cubes are called isomorphic if the are essentially the same. Rotation or reflection of the one cube together with a recoloring of the faces will transform it into the other cube.