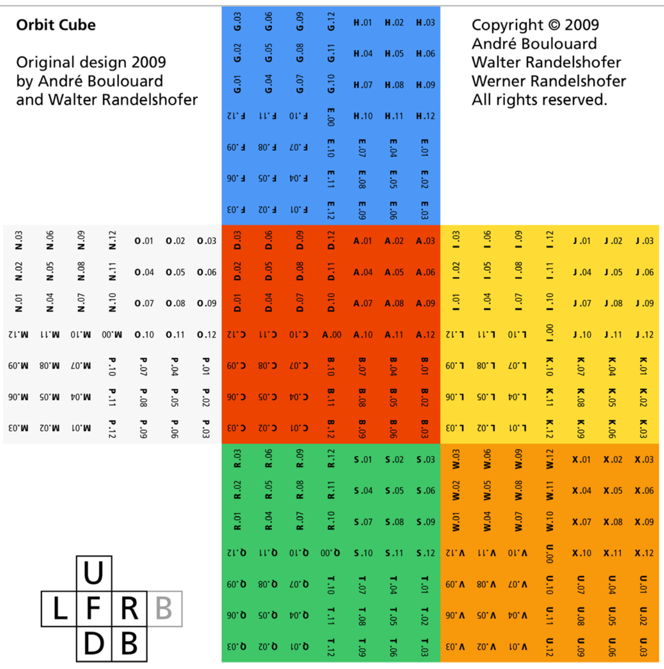
Orbit Cube

Orbit Cube

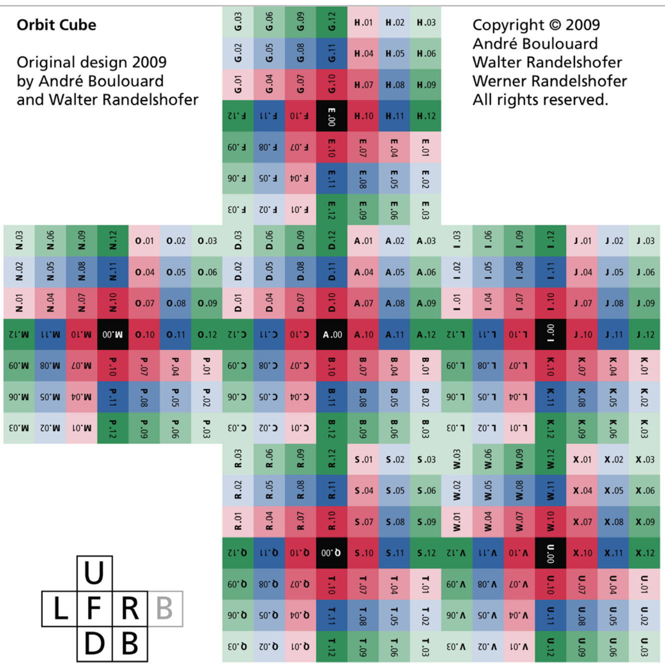
Original design 2009
by André Boulouard
and Walter Randelshofer

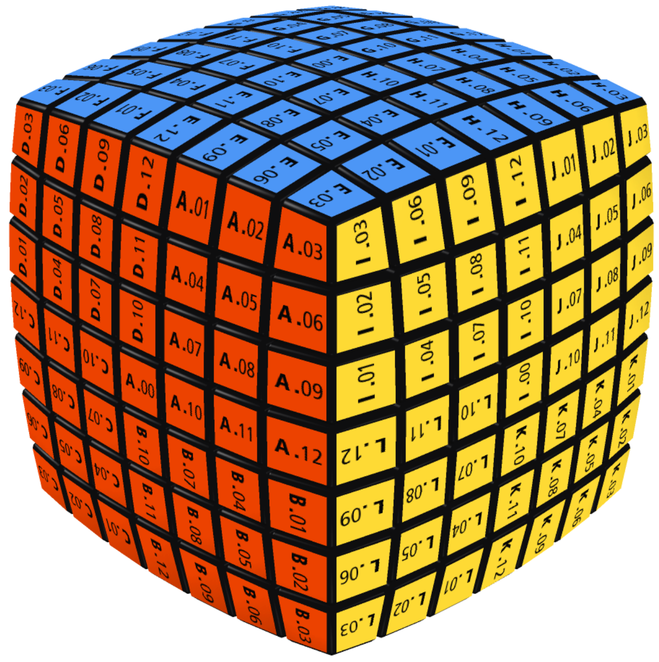


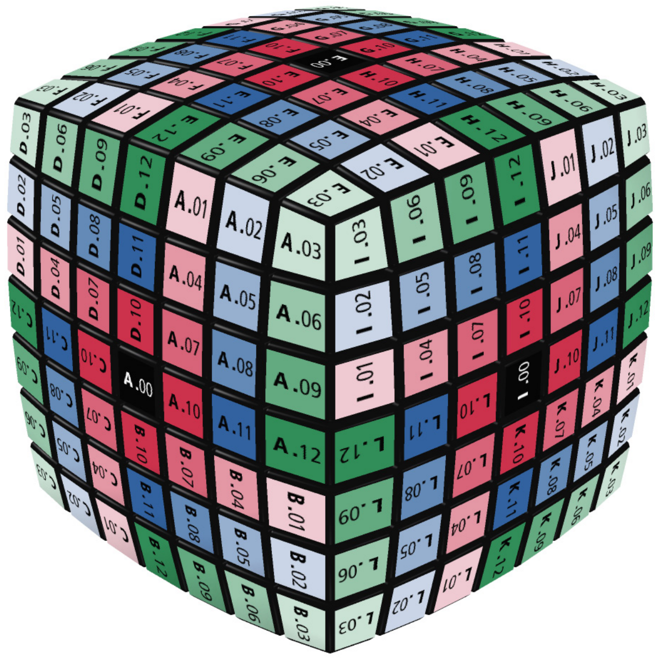
Orbit Cube

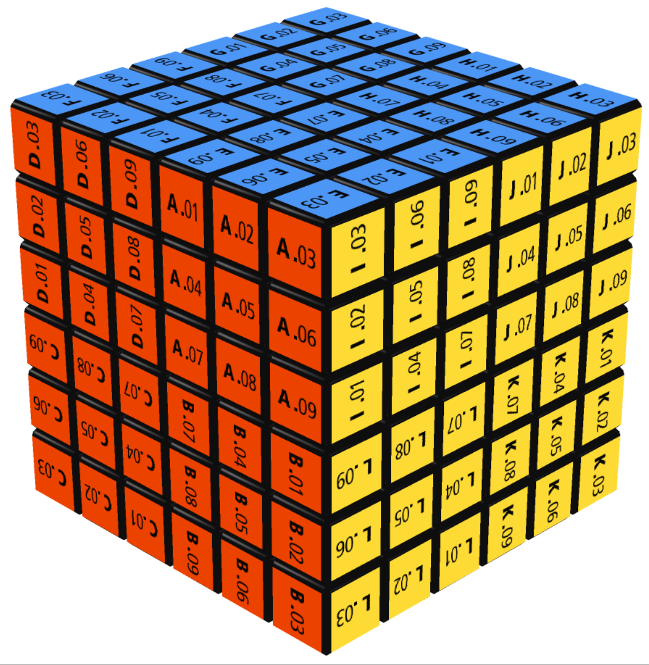
Original design 2009
by André Boulouard
and Walter Randelshofer

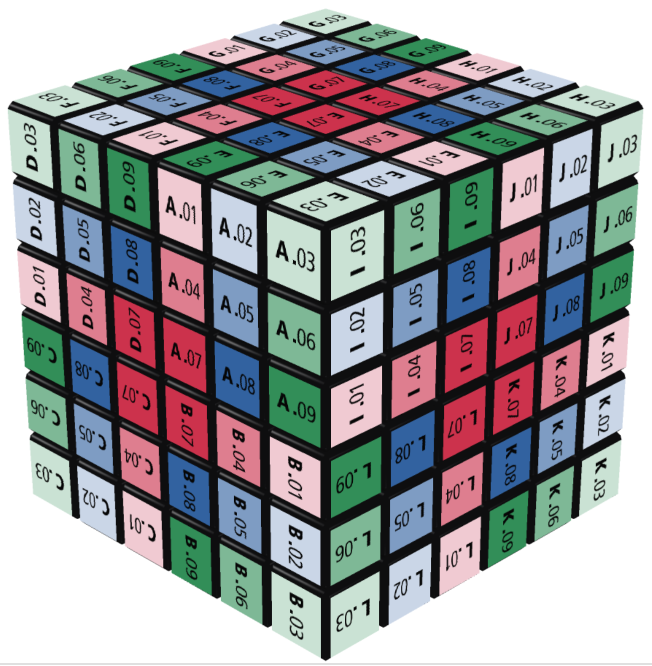
Copyright © 2009
André Boulouard
Walter Randelshofer
Werner Randelshofer
All rights reserved.











Foreword

Document

This document is divided into four main parts that can be read independently.

In **Part I**, basic information is given on cube theory, parity, cycles and orbits. Examples are given to show how to use Orbit Cube textures and n-cycles to permute pieces on a 7x7x7 cubes, using pre-computed algorithms.

Details about how to compute an efficient algorithm database, using templates, are given in **Part II**. Algorithms have been generated to permute pieces as 3-cycles, pairs of 2-cycles, twist and swap corners and fix edge and corner parities, anywhere on any big cube of any order.

Part III is more about database indexing and shows how to compute a fully scalable indexing system.

Examples of n-cycles synthesized with the *Algorithm Picke*r computer program are shown in **Part IV**.

And finally, a scalable cube numbering system is described in the **Annex**.

Links

Since Verdes first released their V-Cube 6 and V-Cube 7, considerable interest has grown in the cubing community in techniques and methods to solve big cubes. Many useful information about these cubes can be found in the following sites:

Big Cubes – Useful Links	
http://www.randelshofer.ch/	http://www.stefan-pochmann.de/
http://cubefreak.net/	http://www.jaapsch.net/puzzles/
http://www.speedcubing.com/chris/	http://www.bigcubes.com/
http://www.speedsolving.com/wiki/	http://www.cubestation.co.uk/
http://michael-gottlieb.blogspot.com/	http://www.v-cubes.com/
Forums	
http://twistypuzzles.com/	http://www.speedsolving.com/
Download CubeTwister	
http://www.randelshofer.ch/cubetwister/	

Contents

To know more…

For users of Microsoft Excel 2007, the AlgorithmPicker7 VBA code can be freely downloaded by clicking the link:

http://www.mementoslangues.fr/CubeDesign/AlgorithmFinder/AlgorithmPicker7.xlsm

For further reading on texture design for virtual cubes, click the link below and select 'CubeDesign':

http://www.mementoslangues.fr/

For any suggestion or comment about this document, please contact me at:

ml (at) mementoslangues.com

I hope you will enjoy reading the document and practicing with virtual Orbit Cubes!

Introduction

This document is about the design of a scalable algorithm database, applicable to any NxNxN cube or supercube, and how it can be used together with a specially designed texture.

A fully scalable database has been computed for a 7x7x7 cube, from which algorithms for cubes of any order can be inferred.

Features

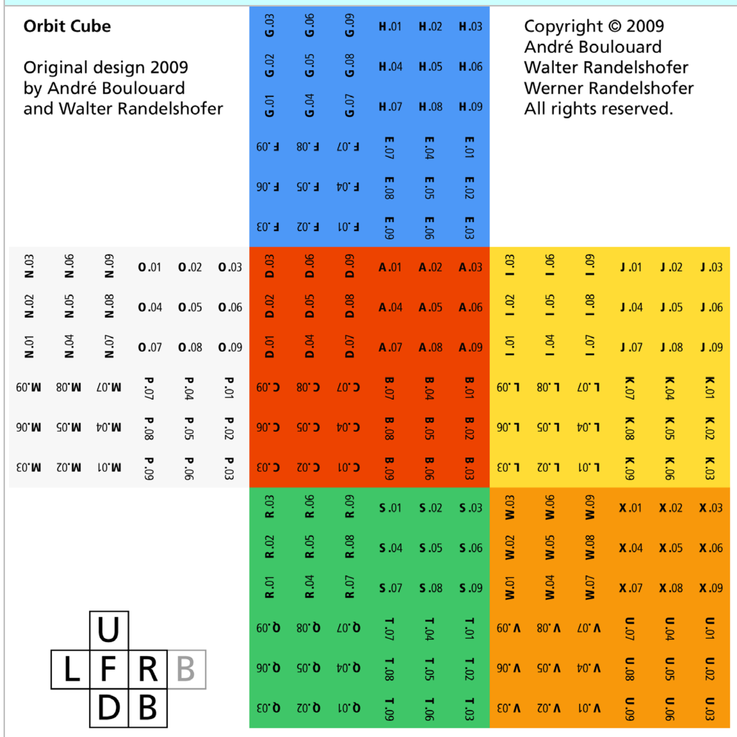
Algorithm databases have been generated for:

1- 3-cycles of corners
2- 3-cycles of edges
3- 3-cycles of centers
4- pairs of 2-cycles of corners (corner twists)
5- 2-cycles of corners (corner swaps – corner parity fix)
6- pairs of 2-cycles of midges (midge flips)
7- 2-cycles of edges (edge flips – edge parity fix)
8- true-center twists

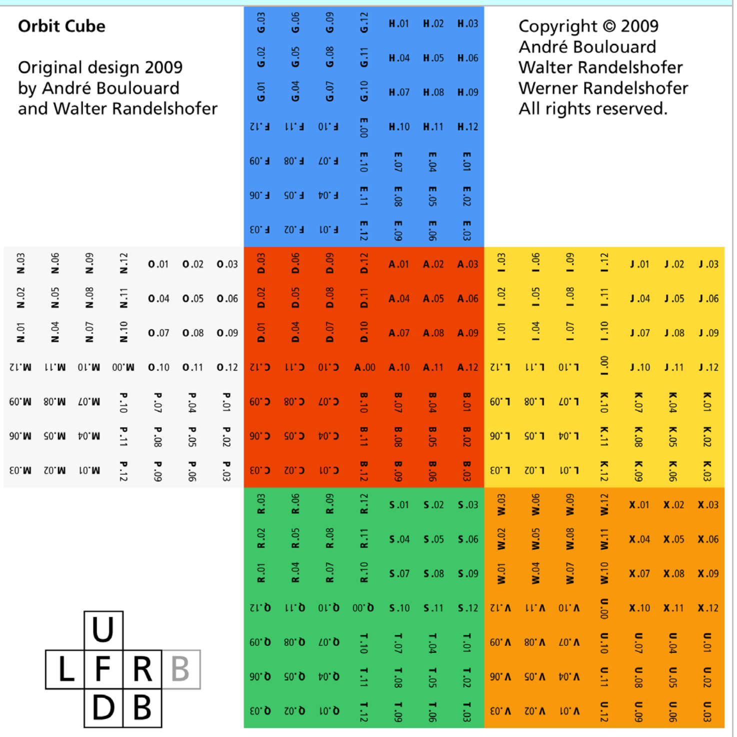
Orbit Cube Textures

By using one of the two textures below on a 6x6x6 or 7x7x7 cube, it becomes easier to see to which orbit a sticker belongs to. Each sticker is marked with an unique identifier tag consisting of a letter from 'A' to 'X' together with an orbit number, which varies from '00' up to '12'. This feature is also useful to highlight n-cycles on a big cube. For example, a 3-cycle of corners on face F would simply show up as cycle (D → A → B → D) in orbit '03'.

Orbit Cube Textures


Orbit Cube
Original design 2009 by André Boulouard and Walter Randelshofer
Copyright © 2009 André Boulouard Walter Randelshofer Werner Randelshofer All rights reserved.
U
L F R B
D B

6x6x6 Cube Texture


Orbit Cube
Original design 2009 by André Boulouard and Walter Randelshofer
Copyright © 2009 André Boulouard Walter Randelshofer Werner Randelshofer All rights reserved.
U
L F R B
D B

7x7x7 Cube Texture

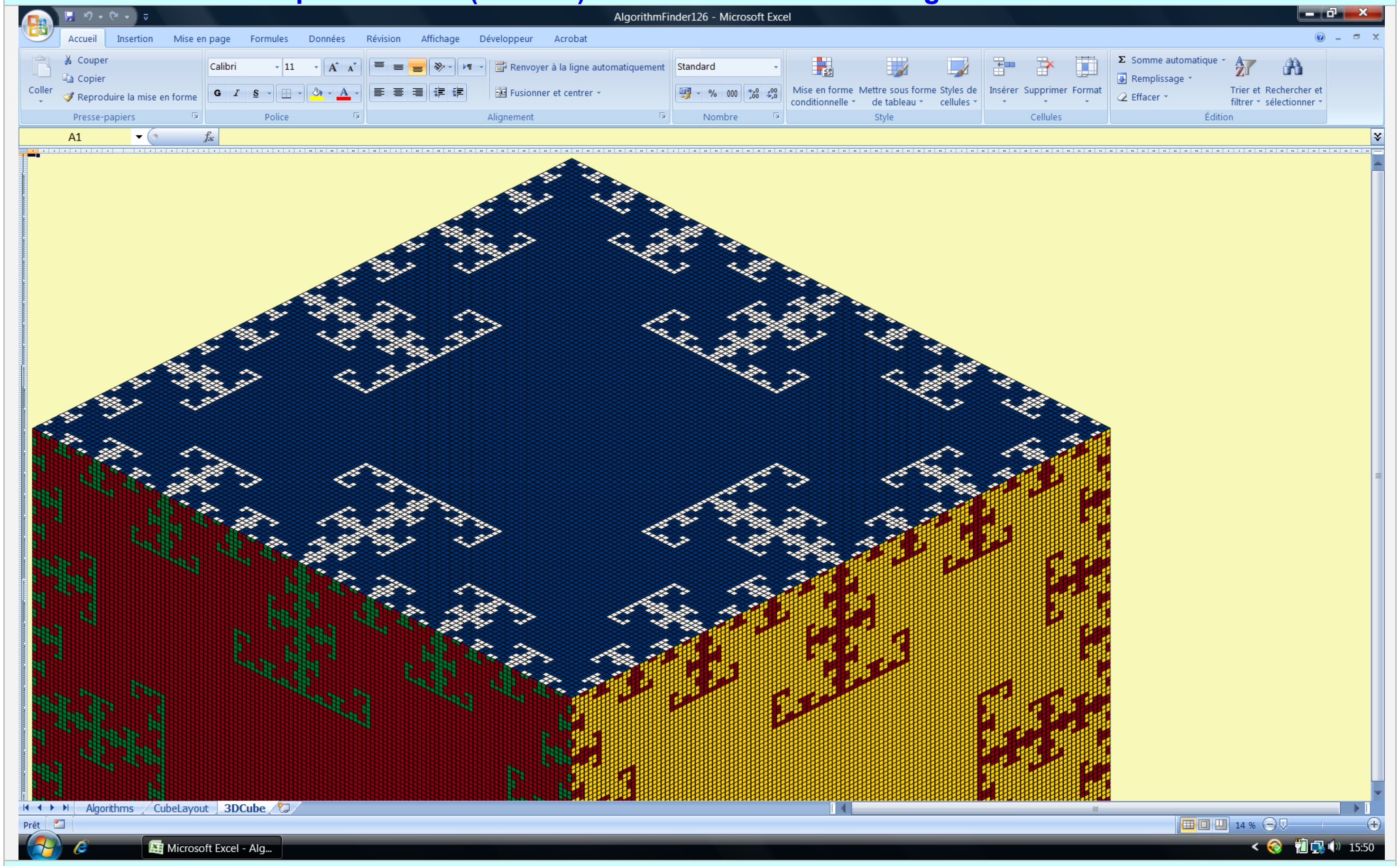
Algorithm Picker

Given an orbit and an n-cycle, the algorithm that will move the n pieces in the orbit can be looked up using an *Algorithm Picker*. This is simply the combination of a piece of software and a UserForm, which is used to query the algorithm database.

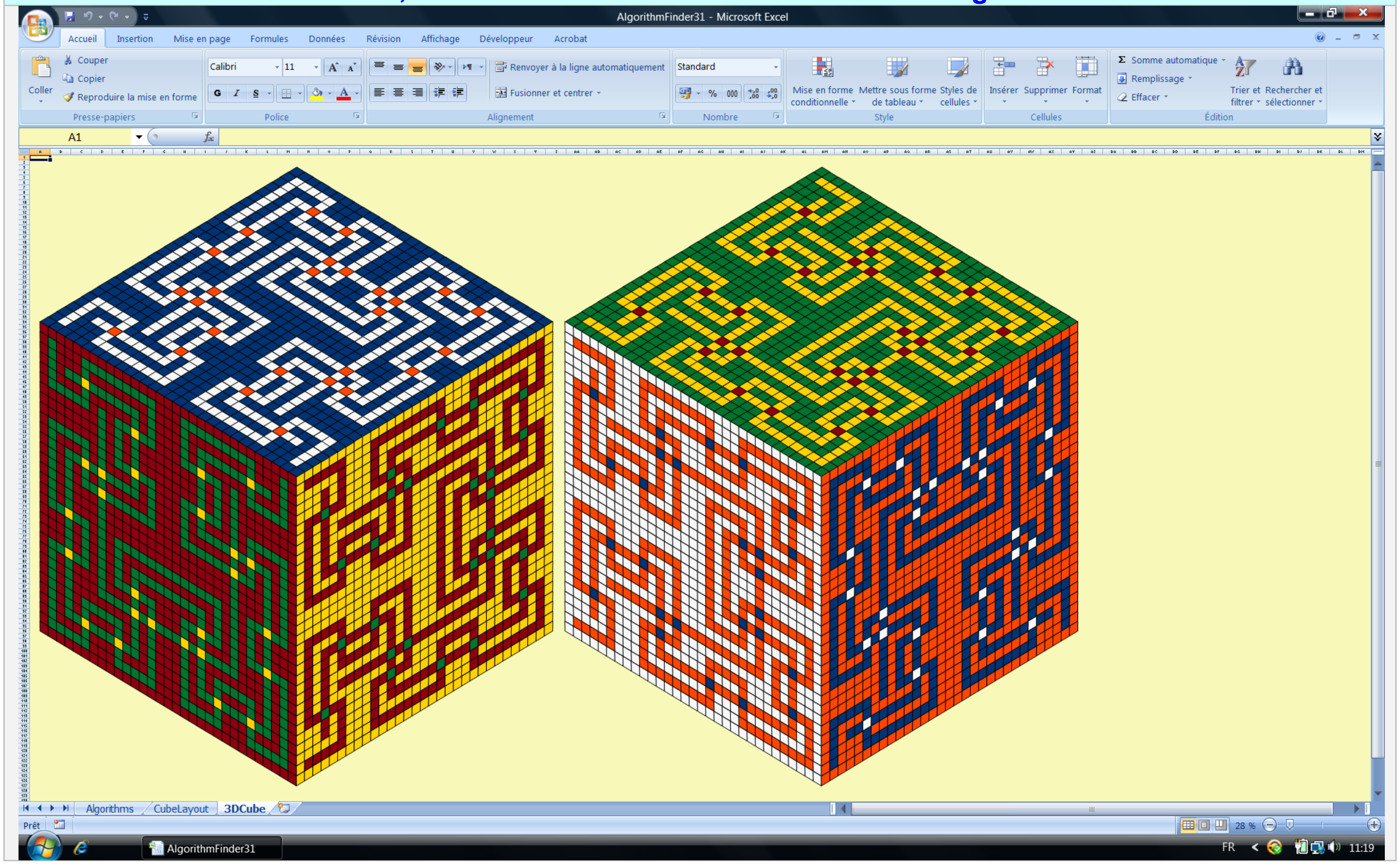
Further Capabilities

Although not directly linked to the Orbit Cube design, I thought it may have been of interest to see how computer program *Algorithm Finder* can be used to display fairly complex pretty patterns (see FractalCubeDesign), because *Algorithm Finder* and *Algorithm Picker* actually share the same N-scalable 'kernel' of subroutines.

T-Square Fractal (Order 6) – 126x126x126 Cube – AlgorithmFinder126



Hilbert Curves, First to Third orders – 31x31x31 Cube – AlgorithmFinder31



Part I

Cube Theory

(From Jaap's Puzzle Page)

Permutations

Suppose you have a list of ordered numbers such as (1,2,3,4,5,6,7,8). A **permutation** of these numbers is simply another list of the same numbers but ordered differently, for example (4,2,6,1,3,5,8,7). Every number on the list must be used exactly once.

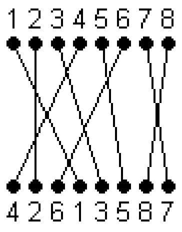
At first sight this is not related to the cube, but suppose you write down the numbers of the eight corner pieces of the cube in a list. Any face move on the cube then permutes corner pieces, and therefore corresponds to a new list with the same numbers of corner pieces but in a different order. Any face move (or move sequence) is therefore a permutation of corner pieces. Of course, the same can be said of edge or center pieces.

By examining what permutations can do, we can therefore examine how pieces of the cube do move. The numbers in our permutation list which are not really important. What is important is how they have moved in the list. A permutation usually uses numbers, but these numbers can represent any items, for example moving pieces of the cube, or any other objects that can be rearranged. A permutation therefore embodies only the movement of items.

There are many different ways to write down a permutation. A common way is to write the original list of numbers on one line, and the new list directly on another line just below. For the list of numbers above, we get:

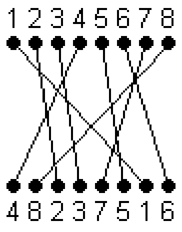
```
1 2 3 4 5 6 7 8
4 2 6 1 3 5 8 7
```

To make this more visual, we can make a line diagram by drawing straight lines between numbers of the two lists to show exactly how each item on the list moves.

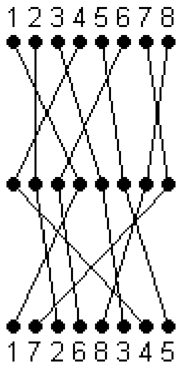


In this example lines cross 8 times, so we can say that 8 is the diagram **crossing number**.

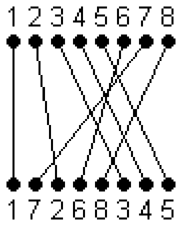
An important aspect of permutations is that they can be combined. On the cube for example, one sequence of moves can be followed by another. In other words, the pieces are rearranged in one way, and starting from the new position they can be rearranged in another way, and so on. Suppose we combine the permutation above with the following permutation, where the crossing number is 15:



Only the movement is important, so even though this second permutation is also written using numbers from 1 to 8, when we combine the two permutations, we only look at the lines, i.e. at what movement is depicted by the permutation. To combine them, we draw the line diagrams one below the other and follow the lines:



By straightening the lines we get:



Let’s now look at crossing numbers. The first two permutations have crossing numbers 8 and 15. The combined drawing therefore has 15+8=23 crosses but when the lines are straightened, we get a crossing number of 11. If you think of the lines as loose threads, and that you physically untangle them, you will see that each time you uncross two threads, the number of crossings always decreases by 2. When you untangle all the threads, the number of crossings then decreases by a multiple of 2. Therefore, although the final permutation crossing number is not simply the sum of the two, it will have the same parity as that sum (i.e. will also be even or also be odd).

Permutation Parity

Let’s call a permutation odd if its crossing number is odd, and even if its crossing number is even. This is the **parity of the permutation**.

It is now easy to see that, when permutations are combined, their parities follow the same rules as numbers:

odd + odd = even

even + even = even

odd + even = odd

even + odd = odd

On a Rubik's Cube, taking into account corners *and* edges, a single quarter turn of a face is an even permutation. To see this, number corners of the face 1-4 and edges 5-8. A quarter turn is then represented by the permutation (2,3,4,1,6,7,8,5) which has crossing number 6 if you draw it.

Combining even permutations will always give another even permutation, so only even permutations of the pieces of the Rubik's Cube are possible. This shows that it is impossible to swap only two pieces (edges or corners) without moving anything else. Note that this holds only for a 3x3x3 cube. Nothing has been said so far about higher order cubes. It can be shown actually that 2 edges can be swapped on a 4x4x4 cube, without disturbing any other piece on the cube.

Disjoint Cycle Notation

A different but very useful notation for permutations is **disjoint cycle notation**. In this notation, the first permutation we used becomes (1 4) (3 5 6) (7 8). This means that 1 moves to the position that 4 was in, and 4 moves to the position that 1 was in. Piece 3 moves to the position that 5 was in, 5 moves to where 6 was, and 6 moves to where 3 used to be. Pieces 7 and 8 swap places just like pieces 1 and 4 do. Piece 2 does not move because it is not mentioned, although you could state so explicitly by including the 2 like this (1 4) (2)(3 5 6) (7 8).

Each parenthesized part of this is called a **cycle**: (1 4) and (7 8) are **2-cycles**, and (3 5 6) is a **3-cycle**. These cycles are called **disjoint** because they use different numbers, and so act on different pieces.

It is easy to see that the parity of a 2-cycle is odd, and that of a 3-cycle is even.

The combined permutation (1 4) (3 5 6) (7 8) is therefore even because odd + even + odd = even. In general the parity of an n-cycle will be the parity of n-1 for any integer number n.

Order of a permutation

The **order of a permutation** is the number of times it has to be performed before the pieces are back to their initial positions. This is where cycle notation is very useful. A 2-cycle is a single swap, so if this is performed twice then the pieces are back in place. A 2-cycle therefore has order 2. Similarly, a 3-cycle will have order 3, and any n-cycle will have order n.

Now let's consider a permutation like (1 4) (3 5 6) (7 8) which is composed of several cycles. If it is performed twice, or in fact any even number of times then the 2-cycles will disappear. If it is performed three times or any multiple of three, then the 3-cycle will disappear. If it is performed 6 times, which is a multiple of both 2 and 3 then all the cycles disappear and this permutation therefore has order 6.

In general, the order of a permutation is the Least Common Multiple (LCM) of the lengths of the disjoint cycles.

Let's see how this applies to the cube. Consider the move sequence F R. If we look at how this moves pieces, we see that the UFL corner moves to position RUB, corner RUB moves to position RBD, and so on. In cycle notation this permutation is written as follows:

(UFL RUB RBD RDF DLF) (UF RU RB RD RF DF LF)

This is a 5-cycle of corners and a 7-cycle of edges. Its order is therefore 35, which means that if you constantly repeat the move sequence F R on the cube, you will have to do it 35 times before the pieces come back to their original positions. If you try this out on a solved cube you will see that although this is true, the cube is not restored because some corners are twisted. We have only looked at the location of the pieces, and not at their orientation.

Groups

The collection of all *possible* permutations of n items form a **group**. A group is simply a collection of things (usually called elements of the group) which satisfy various conditions, which I will list below. We use these conditions implicitly whenever we use permutations, so it is best to state them explicitly now.

Any two elements of a group can be combined, and this results in another element of the group. As you have seen, two permutations can be combined by performing one after the other, and this will always result in a permutation. If P and Q are permutations, then P Q will be the permutation resulting from performing first P and then Q. Combining two elements of a group is usually called multiplication of the two elements.

There is an **identity**, i.e. an element I in the group such that for any element P in the group we have:

$$P\,I = I\,P = P$$

In a permutation group the element I is simply the permutation that does not move anything.

Every element has an **inverse**, i.e. if P is an element of the group then there is an element Q in the group such that:

$$P\,Q = Q\,P = I$$

If you mirror the line diagram of a permutation vertically, you get the line diagram of its inverse. A permutation in cycle notation can be inverted just by writing each cycle in reverse. The inverse of permutation P is denoted by P^{-1}, or by P'. On puzzles, you can do the inverse of a move sequence just by undoing the moves in reverse order, i.e. taking back the moves you did.

The multiplication is **associative**, i.e. if P, Q and R are elements of the group, then:

(P Q) R = P (Q R)

For permutations this is obviously true.

If a group is **commutative**, we have:

P Q = Q P for any P, Q in the group

A commutative group is called an Abelian group. Puzzles that belong to an Abelian group are much simpler to solve because the order in which the moves are performed does not matter. But permutation groups occurring in most puzzles are generally not Abelian.

All possible movements of the pieces on a Rubik's cube also form a group, the **Cube Group**. At first sight this is not a simple permutation group because the orientation of pieces matter, but you could consider it as a permutation group of 48 moving facelets instead of 20 moving pieces.

On a pedantic note, it is technically incorrect to say that the positions of the Rubik's cube form a group. A position is reached from the solved cube state by moving pieces in some way, and it is those movements that form the group because movements can be combined. On the standard Rubik's Cube this is not a very important distinction, but on other puzzles like the 4x4x4 cube it is. Thist puzzle has center pieces which look the same, and so there are positions which are indistinguishable from each other. Thus different permutations seem to correspond to the same position, or permutations which seem to do nothing in one position will change things in another. The permutations still form a group, but the positions do not (unless you mark the centre pieces so they can be distinguished).

Conjugation

If P and Q are elements of a group, then the **conjugate** of Q by P is the element P Q P'. This is one of the most useful concepts for solving a puzzle like the cube. Let’s illustrate this with an example on the Rubik's Cube.

Suppose that you know that the move sequence Q = R' L F2 R L' U2 cycles three edges around, viz. (UB,UF,DF) but that you want to cycle 3 other edges of the cube, for example (UR,UF,UL). We would like to know how to cycle them if they where placed at positions UB, UF and DF, so we simply put them there, for example by doing the sequence P = F2 U. This sequence moves other pieces as well, but that does not matter. So after we put them in position with P, cycled them with Q, we put everything back by doing P'. The relevant edges have been cycled as we wanted, and any other pieces that were moved by P are put back by P'.

Therefore P Q P' = F2 U R' L F2 R L' U F2 cycles just edges (UR,UF,UL).

As you can see from this example, if you have a sequence that performs a certain task on particular pieces of the puzzle, conjugation will allow you to perform the same task on any other similar pieces of the puzzle instead. So, if you can flip two edge pieces then you can flip any two of them, or if you can twist two corners then you can twist any two of them, and so on.

Commutation

Conjugation allowed you to apply a specific sequence more generally, but you still need to find that specific sequence to begin with. That is where **commutation** is useful.

If P and Q are elements of a group, then [P, Q] = P Q P' Q' is called a **commutator**.

If P and Q commute (for example if they are disjoint, like R and L moves on the cube) then:

P Q P' Q' = Q P P' Q' = Q I Q' = Q Q'= I

A commutation can be seen as an indication of whether P and Q commute, and by how much. If P and Q are nearly disjoint, then the commutator will move fairly little, and therefore often performs a useful function when solving a puzzle.

The simplest commutators on a cube use single face moves for P and Q, for example P = F and Q = R' give:

[F, R'] = F R' F' R

This cycles three edges (FU,FR,UR), and two pairs of corners (UFL,BRU) and (URF,RDF). Note that corner UFL moves to BRU, which in turn moves to LUF. This is twisted anti-clockwise compared to the original position FLU, so if we perform this cycle twice, these two pieces will be back to their original positions but will both be twisted anti- clockwise. The other corner 2-cycle of F R' F' R twists clockwise. We could adapt cycle notation to show this as follows:

F R' F' R = (UFL, BRU)- (URF, RDF)+ (FU, FR, UR)

Doing this twice, we get:

(F R' F' R)2 = (UFL)- (UBR)- (URF)+ (DFR)+ (FU, UR, FR)

Doing this three times, we get:

(F R' F' R)3 = (UFL, UBR) (URF, DFR)

Theoretically these moves and their conjugates are enough to perform any even permutation of corners, and any even permutation of edges. Any single quarter turn of a face is an odd permutation of corners plus an odd permutation of edges, so just using what we have so far we could position all pieces of the Cube.

What remains is just to orient them.

Commutation works best when P and Q are nearly disjoint.

Therefore lets choose Q to be a turn of the U face, and P so that it affects only a single piece in the U face. An extremely useful choice is the **monotwist** P = R' D R F D F'. This twists one corner (URF)+ and does not affect anything else in the U layer. The bottom half of the cube is messed up but that does not matter. We now have the following very useful sequences:

[P, U] = P U P' U' = R' D R F D F' U F D' F' R' D' R U' = (URF)+ (UBR)-

[P, U2] = P U2 P' U2 = R' D R F D F' U2 F D' F' R' D' R U2 = (URF)+ (ULB)-

[P, U'] = P U' P' U = R' D R F D F' U' F D' F' R' D' R U = (URF)+ (UFL)-

You can now twist any two corners on the cube.

Other good choices for P are the **monoflip** P = F U D' L2 U2 D2 R U = (FU)+, which will allow you to flip any edge on the cube, and P = R' D R which gives you a simple 3-cycle of corners.

It is also productive to let Q be a move of a middle slice, for example MR. If you look squarely at the R face of the cube, MR is a clockwise quarter turn of the middle slice just behind the R layer.

If we let P = F2, which is a **monoswap** of edges (DF,UF) in the middle slice, then we get a 3-cycle of edges:

[F2, MR] = F2 MR F2 MR' = (DB,DF,UF)

and the 2-H pattern:

[F2, MR2] = F2 MR2 F2 MR2 = (DF,UF)(DB,UB)

If P = MF , then we get the 6-spot pattern:

[MF, MR] = MF MR MF' MR'

whereas P = MF2 gives the 4-spot pattern:

[MF2, MR] = MF2 MR MF2 MR'

Another good choice is P = F U' R F' U, a neat **monoflip** of edges (FU)+:

[F U' R F' U, MR] = F U' R F' U MR U' F R' U F' MR'

Size of the group

You may have noticed that when you use a commutator for twisting corners, you will always twist two corners in opposite directions. Commutators can also only flip pairs of edges. It turns out that this is enough to solve the cube because it is impossible for a single piece to be turned.

Every corner piece has one facelet that belongs in the U or the D face. For a corner which has been moved anywhere on the cube, lets define its twist as follows:

1- Its twist is 0 if its U/D facelet is in the U or D face.
2- Its twist is +1 if the piece has been turned clockwise from the 0 twist orientation.
3- Its twist is -1 if the piece has been turned anti-clockwise from the 0 twist orientation.

Twisting a corner clockwise will increase its twist by one. We have to work modulo 3 however, because 3 twists is the same as no twist at all, and so a twist value of +2 is really only a twist of +2-3 = -1. Similarly, an anti-clockwise twist decreases the twist by one modulo 3 (i.e. a twist of -2 is just a twist of +1).

If you turn the U or the D face, the twists of corner pieces do not change. If you turn any other face a quarter turn, then the twist of two of the corner pieces increases, and the twist of the other two corners decreases. In any case the total twist of all the corners does not change modulo 3. In the starting position the cube has a total twist of 0, and this therefore remains 0 however mixed up the cube gets. This shows that it is impossible to twist a single corner in isolation, and that if you twist only two corners then they must go in opposite directions.

A very similar method can be used for the edges. Define an edge flip as 0 or 1 (modulo 2) and show that the total flip remains equal to 0, whatever move is performed, which means that no edge can be flipped in isolation. This holds for the Rubik's Cube only: it can be shown that an isolated dedge can be flipped on a 4x4x4 cube.

Another way is by looking at the permutations of edge facelets. A quarter turn of a face is an even permutation of edge facelets (two 4-cycles), so that any move sequence will give only even permutations of edge facelets.

A single edge flip is an odd permutation of edge facelets on a Rubik's Cube and hence not possible without taking the cube apart.

In all we have now seen three restrictions on the possible layout of pieces on a Rubik's Cube:

1- The total corner twist must be eqaul to zero modulo 3
2- The total edge flip must be equal to zero modulo 2
3- The parity of pieces permutation must be even

If you were to take the cube apart and randomly put it back together again, there would be a 1 in 3 chance of having the right corner twist (since all three possible twist values are equally likely). Similarly there is a 1 in 2 chance to get the total edge flip correct, and a 1 in 2 chance of getting the right permutation parity. Putting this together, we find there is a 1 in 12 chance that a randomly assembled cube is solvable.

Sticker Orbits

If a *sticker* orbit is defined as any complete set of stickers that can move into each other's position, then the total number of stickers per orbit is 24 for corner, edges and non-true centers (true centers are only found on odd-order cubes).

Note that there are 3 stickers per corner piece, 2 stickers per edge piece and 1 sticker per center piece.

The total number of sticker orbits for any NxNxN cube, excluding true centers stickers, can be calculated by noting that there are:

1- 1 corner orbit
2- $Ne = N - 2$ edge orbits
3- $Nc = (p - 1)^2$ center orbits, where $p = N/2$ is an integer number, (even-order cubes only)
4- $Nc = (p - 1).p$ center orbits, where $p = (N - 1)/2$ is an integer number, (odd-order cubes only

This shows that the number of center orbits is *always* odd for odd-order cubes and may be even or odd for even-order cubes, depending on N.

For even-order cubes, the number of center orbits Nc is odd if and only if cube order $N_{\text{Even-Order Cube}}$ is given by:

$$N_{\text{Even-Order Cube}} = 2.(\text{AnyOddNumber} + 1)$$

This means that the number of center orbits is *odd* for the following cube orders:

$$N_{\text{Even-Order Cube}} = 4, 8, 12, 16, 20, 24,\ldots$$

Conversely, the number of center orbits is *even* for cube orders from the following list:

$$N_{\text{Even-Order Cube}} = 6, 10, 14, 18, 22, 26,\ldots$$

From this, the total number of sticker orbits has been calculated for a number of NxNxN cubes and is shown in the table below:

Total Number of Sticker Orbits										
	Odd-Order Cubes: $(N^2 - 1)/4$					**Even-Order Cubes: $N^2/4$**				
Cube Order N	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**20**
Corner Orbit	1	1	1	1	1	1	1	1	1	1
Edge Orbits	1	2	3	4	5	6	7	8	9	18
Center Orbits	0	1	2	4	6	9	12	16	20	81
Total	**2**	**4**	**6**	**9**	**12**	**16**	**20**	**25**	**30**	**100**

From this table, we can infer a *very* crude approximation of the *average* number of moves that would be needed to *optimally* solve any NxNxN cube, under the following assumptions:

1- There is a *single* 3-cycle of stickers in each orbit, on average
2- Each 3-cycle is 10 moves long, on average

This would give an average number of moves of 20 to solve a 3x3x3 cube, 40 for the 4x4x4, 60 for the 5x5x5, 90 for the 6x6x6 and 120 for the 7x7x7.

Sticker Permutation Vector

There is a total of $6.N^2$ stickers on any NxNxN cube. If these are numbered from 0 to $6.N^2 – 1$, then the cube state can be represented by a permutation vector of size $6.N^2$. In the initial state, the permutation vector is an *ordered* set of $6.N^2$ integer numbers:

Sticker permutation vector (initial state) = $(0, 1, 2,\dots, 6.N^2 – 1)$

After a number of outer layer and/or inner slice turns, the new permutation vector is no longer an ordered set of numbers because some have moved to new locations. This can be shown on an example. Let’s take the case of a single 3-cycle of centers on a 7x7x7 cube, where sticker 8 has moved to location 159, sticker 159 to location 12 and sticker 12 to location 8 ($8 \rightarrow 159 \rightarrow 12 \rightarrow 8$):

Sticker permutation vector (initial state) = (0, 1, 2,…, **8**,…, **12**,…, **159**,…, 293)

Sticker permutation vector (new state) = (0, 1, 2,…, **12**,…, **159**,…, **8**,…, 293)

algorithm (3-cycle of centers) = F' NL NF' NL' F NL NF NL' = [F', NL NF' NL']

The two permutation vectors differ only by 3 numbers. All other stickers stay in their initial locations.

Cycle Decomposition

(For more on cycles and permutations, see: Permutation)

Decomposing Cycles

Any permutation of stickers in a given orbit can be decomposed into a number of disjoint 2- and 3-cycles. Some examples of cycle decomposition are shown below for cycles of length 4 to 12:

Length 4: (1 2 3 4) = (1 2 3) (1 4) = (1 2) (1 3 4)

Length 5: (1 2 3 4 5) = (1 2 3) (1 4 5)

Length 6: (1 2 3 4 5 6) = (1 2 3) (1 4 5 6) = (1 2 3) (1 4 5) (1 6)

Length 7: (1 2 3 4 5 6 7) = (1 2 3) (1 4 5 6 7) = (1 2 3) (1 4 5) (1 6 7)

Length 8: (1 2 3 4 5 6 7 8) = (1 2 3) (1 4 5 6 7 8) = (1 2 3) (1 4 5) (1 6 7 8) = (1 2 3) (1 4 5) (1 6 7) (1 8)

Length 9: (1 2 3 4 5 6 7 8 9) = (1 2 3) (1 4 5 6 7 8 9) = (1 2 3) (1 4 5) (1 6 7 8 9) = (1 2 3) (1 4 5) (1 6 7) (1 8 9)

Length 10: (1 2 3 4 5 6 7 8 9 10) = (1 2 3) (1 4 5 6 7 8 9 10) = (1 2 3) (1 4 5) (1 6 7 8 9 10) =
(1 2 3) (1 4 5) (1 6 7) (1 8 9 10) = (1 2 3) (1 4 5) (1 6 7) (1 8 9) (1 10)

Length 11: (1 2 3 4 5 6 7 8 9 10 11) = (1 2 3) (1 4 5 6 7 8 9 10 11) = (1 2 3) (1 4 5) (1 6 7 8 9 10 11) =
(1 2 3) (1 4 5) (1 6 7) (1 8 9 10 11) = (1 2 3) (1 4 5) (1 6 7) (1 8 9) (1 10 11)

Length 12: (1 2 3 4 5 6 7 8 9 10 11 12) = (1 2 3) (1 4 5 6 7 8 9 10 11 12) = (1 2 3) (1 4 5) (1 6 7 8 9 10 11 12) =
(1 2 3) (1 4 5) (1 6 7) (1 8 9 10 11 12) = (1 2 3) (1 4 5) (1 6 7) (1 8 9) (1 10 11 12) =
(1 2 3) (1 4 5) (1 6 7) (1 8 9) (1 10 11) (1 12)

Note that there is more than one way to decompose any n-cycle into disjoint 3- or 2-cycles.

Recomposing Cycles

Any *pair* of 2-cycles of stickers in a given orbit can be recomposed into a pair of 3-cycles. Here is how to proceed:

1- Select a first 2-cycle and a second 2-cycle.
2- Select the first number of the first 2-cycle and the last number of the second 2-cycle.
3- Insert the last number at the end of the first 2-cycle and the first number at the beginning of the second 2-cycle.
4- Continue until all pairs of 2-cycles have been transformed into pairs of 3-cycles.

This shows that pairs of 2-cycles can be recomposed into pairs of 3-cycles. This means also that any orbit state can be reached using only 3-cycles, provided that the permutation leading to this state if of even parity.

Examples below show how cycles can be decomposed/recomposed at will. Note again that there is no unique solution.

Even length cycles

Two 2-cycles (= Two 3-cycles)

(1 2) (3 4) = (1 2 4) (1 3 4)

Two 4-cycles (= Four 3-cycles)

(1 2 3 4) (5 6 7 8) = (1 2 3) (1 4) (5 6) (5 7 8) = (1 2 3) (1 4 6) (1 5 6) (5 7 8)

Two 6-cycles (= Six 3-cycles)

(1 2 3 4 5 6) (7 8 9 10 11 12) = (1 2 3) (1 4 5) (1 6 8) (1 7 8) (7 9 10) (7 11 12)

Two 8-cycles (= Eight 3-cycles)

($\underline{1}$ 2 3 4 5 6 7 8) ($\underline{9}$ 10 11 12 13 14 15 16) = ($\underline{1}$ 2 3) ($\underline{1}$ 4 5) ($\underline{1}$ 6 7) ($\underline{1}$ 8 10) ($\underline{1}$ $\underline{9}$ 10) ($\underline{9}$ 11 12) ($\underline{9}$ 13 14) ($\underline{9}$ 15 16)

Two 10-cycles (= Ten 3-cycles)

($\underline{1}$ 2 3 4 5 6 7 8 9 10) ($\underline{11}$ 12 13 14 15 16 17 18 19 20) =
($\underline{1}$ 2 3) ($\underline{1}$ 4 5) ($\underline{1}$ 6 7) ($\underline{1}$ 8 9) ($\underline{1}$ 10 12) ($\underline{1}$ $\underline{11}$ 12) ($\underline{11}$ 13 14) ($\underline{11}$ 15 16) ($\underline{11}$ 17 18) ($\underline{11}$ 19 20)

Two 12-cycles (= Twelve 3-cycles)

($\underline{1}$ 2 3 4 5 6 7 8 9 10 11 12) ($\underline{13}$ 14 15 16 17 18 19 20 21 22 23 24) =
($\underline{1}$ 2 3) ($\underline{1}$ 4 5) ($\underline{1}$ 6 7) ($\underline{1}$ 8 9) ($\underline{1}$ 10 11) ($\underline{1}$ 12 14) ($\underline{1}$ $\underline{13}$ 14) ($\underline{13}$ 15 16) ($\underline{13}$ 17 18) ($\underline{13}$ 19 20) ($\underline{13}$ 21 22) ($\underline{13}$ 23 24)

Four 2-cycles (= Four 3-cycles)

($\underline{1}$ 2) (3 $\underline{4}$) ($\underline{5}$ 6) (7 $\underline{8}$) = ($\underline{1}$ 2 $\underline{4}$) ($\underline{1}$ 3 $\underline{4}$) ($\underline{5}$ 6 $\underline{8}$) ($\underline{5}$ 7 $\underline{8}$)

Four 4-cycles (= Eight 3-cycles)

($\underline{1}$ 2 3 4) ($\underline{5}$ 6 7 8) ($\underline{9}$ 10 11 12) ($\underline{13}$ 14 15 16) =
($\underline{1}$ 2 3) ($\underline{1}$ 4 6) ($\underline{1}$ $\underline{5}$ 6) ($\underline{5}$ 7 8) ($\underline{9}$ 10 11) ($\underline{9}$ 12 14) ($\underline{9}$ $\underline{13}$ 14) ($\underline{13}$ 15 16)

Four 6-cycles (= Twelve 3-cycles)

($\underline{1}$ 2 3 4 5 6) ($\underline{7}$ 8 9 10 11 12) ($\underline{13}$ 14 15 16 17 18) ($\underline{19}$ 20 21 22 23 24) =
($\underline{1}$ 2 3) ($\underline{1}$ 4 5) ($\underline{1}$ 6 8) ($\underline{1}$ $\underline{7}$ 8) ($\underline{7}$ 9 10) ($\underline{7}$ 11 12) ($\underline{13}$ 14 15) ($\underline{13}$ 16 17) ($\underline{13}$ 18 20) ($\underline{13}$ $\underline{19}$ 20) ($\underline{19}$ 21 22) ($\underline{19}$ 23 24)

One 2-cycle + one 4-cycle (= Three 3-cycles)

($\underline{1}$ 2) (3 4 5 $\underline{6}$) = ($\underline{1}$ 2) (3 4) (3 5 $\underline{6}$) = ($\underline{1}$ 2 4) ($\underline{1}$ 3 4) (3 5 $\underline{6}$)

One 4-cycle + one 2-cycle (= Three 3-cycles)

($\underline{1}$ 2 3 4) (5 $\underline{6}$) = ($\underline{1}$ 2 3) ($\underline{1}$ 4) (5 $\underline{6}$) = ($\underline{1}$ 2 3) ($\underline{1}$ 4 $\underline{6}$) ($\underline{1}$ 5 $\underline{6}$)

Odd Length Cycles

5-cycle (= Two 3-cycles)

($\underline{1}$ 2 3 4 5) = ($\underline{1}$ 2 3) ($\underline{1}$ 4 5)

(1 2 3 4 5) = (1 2 5) (3 4 5)

(1 2 3 4 5) = (1 4 5) (2 3 4)

(1 2 3 4 5) = (2 3 4) (1 2 5)

(1 2 3 4 5) = (3 4 5) (1 2 3)

7-cycle   (= Three 3-cycles)

($\underline{1}$ 2 3 4 5 6 7) = ($\underline{1}$ 2 3) ($\underline{1}$ 4 5 6 7) = ($\underline{1}$ 2 3) ($\underline{1}$ 4 5) ($\underline{1}$ 6 7)

9-cycle (= Four 3-cycles)

($\underline{1}$ 2 3 4 5 6 7 8 9) = ($\underline{1}$ 2 3) ($\underline{1}$ 4 5 6 7 8 9) = ($\underline{1}$ 2 3) ($\underline{1}$ 4 5) ($\underline{1}$ 6 7 8 9) = ($\underline{1}$ 2 3) ($\underline{1}$ 4 5) ($\underline{1}$ 6 7) ($\underline{1}$ 8 9)

11-cycle (= Five 3-cycles)

($\underline{1}$ 2 3 4 5 6 7 8 9 10 11) = ($\underline{1}$ 2 3) ($\underline{1}$ 4 5) ($\underline{1}$ 6 7) ($\underline{1}$ 8 9) ($\underline{1}$ 10 11)

13-cycle (= Six 3-cycles)

(1 2 3 4 5 6 7 8 9 10 11 12 13) = (1 2 3) (1 4 5) (1 6 7) (1 8 9) (1 10 11) (1 12 13)

15-cycle (= Seven 3-cycles)

(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15) = (1 2 3) (1 4 5) (1 6 7) (1 8 9) (1 10 11) (1 12 13) (1 14 15)

17-cycle (= Eight 3-cycles)

(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17)
= (1 2 3) (1 4 5) (1 6 7) (1 8 9) (1 10 11) (1 12 13) (1 14 15) (1 16 17)

19-cycle (= Nine 3-cycles)

(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19)
= (1 2 3) (1 4 5) (1 6 7) (1 8 9) (1 10 11) (1 12 13) (1 14 15) (1 16 17) (1 18 19)

21-cycle (= Ten 3-cycles)

(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21)
= (1 2 3) (1 4 5) (1 6 7) (1 8 9) (1 10 11) (1 12 13) (1 14 15) (1 16 17) (1 18 19) (1 20 21)

23-cycle (= Eleven 3-cycles)

(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23)
= (1 2 3) (1 4 5) (1 6 7) (1 8 9) (1 10 11) (1 12 13) (1 14 15) (1 16 17) (1 18 19) (1 20 21) (1 22 23)

Optimized algorithms

As all 3-cycles may differ in number of moves, a way of optimizing (ie. minimizing) the length of a *compound* algorithm would be to select the shortest algorithm between all combinations.

Let's consider a 15-cycle for example. There are at least fifteen ways of building a 15-cycle from seven 3-cycles. By running *all* fifteen combinations and selecting the *shortest* one, a reduction in number of moves of the compound algorithm will result. As the length of 3-cycles is usually >= 8 and <= 12, by doing so the resulting optimized algorithm will usually be 8 moves long.

Number of 3-Cycles

Edges and Centers

Taking the ordering of sticker numbers into account, the maximum number of 3-cycles in an orbit of 24 stickers of edges or centers is given by the number of arrangements of k = 3 stickers among n = 24 stickers, without repetitions:

$$A_{n,k} = n!/(n - k)! = A_{24,3} = 12144$$

This means that a maximum of 12144 algorithms would be necessary for permuting stickers as triplets using only 3-cycles for an orbit of 24 stickers. If we already know the total number of center and edge orbits for an NxNxN cube, then we can compute the maximum number of algorithms needed to solve centers and edges using only 3-cycles:

Maximum Number of Algorithms – 3-Cycles of Centers or Edges

N	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**20**
Orbits	1	3	5	8	11	15	19	24	29	99
Algs	**12 144**	**36 432**	**60 720**	**97 152**	**133 584**	**182 160**	**230 736**	**291 456**	**352 176**	**1 202 256**

Note that due to edge move restrictions, the actual number of algorithms is less than shown above.

Corners

Taking the ordering of sticker numbers into account, the maximum number of 3-cycles in an orbit of 24 stickers of corners is given by:

$$24.21.18 = 9072$$

This means that a maximum of 9072 algorithms would be necessary for permuting stickers as triplets using only 3-cycles for an orbit of 24 stickers. Note again that due to corner move restrictions, the actual number of algorithms is less than that.

Corners 3-cycle

In a 3-cycle of corners, three pieces are permuted in such a way that:

1- the parity of the permutation of pieces is even, and
2- the sum of corner twists, due to the permutation, is equal to zero (Modulo 3)

For a 3-cycle, the first condition is already met because all three pieces are different from each other. In fact, any permutation of an odd number of pieces, ie. 3, 5, 7, etc., is *always* of even parity. Note that if the first condition is met, then the second condition is *automatically* met. There are just 9 different cases of corner twists:

1- all three corner twists are equal (3 cases), or
2- two twists out of three add to zero and the remaining twist is equal to zero (6 cases)

If a counterclockwise (CCW) twist is represented by '-1', a no twist by '0' and a clockwise (CW) twist by '+1', then the 9 cases are as shown in the table below:

Corners 3-cycle – Corner Twists – The 9 Cases				
Sticker Permutation Exemple	Corner 0	Corner 1	Corner 2	Sum Modulo 3
(D A B)	0	0	0	0
(D E B)	0	+1	-1	0
(D I B)	0	-1	+1	0
(O A B)	+1	-1	0	0
(O E B)	+1	0	-1	0
(O I B)	+1	+1	+1	0
(F A B)	-1	+1	0	0
(F E B)	-1	-1	-1	0
(F I B)	-1	0	+1	0
Corner Permutation: (Corner 0 → Corner 1 → Corner 2 → Corner 0)				

If stickers are used instead of 'pieces' and 'twists', there are 27 possible cases for a given 3-cycle of corner pieces, of which only 9 are really different cases. For example, let's consider the first three corners of the Orbit Cube, where each corner is defined as a group of 3 stickers:

1- Corner 0 is (D, O, F)
2- Corner 1 is (A, E, I)
3- Corner 2 is (B, L, S)
4- Corner permutation is defined as: (Corner 0 → Corner 1 → Corner 2 → Corner 0)
5- Corner twists are unspecified so that *all* possible cases will be considered

Corner Numbering & Lettering – Orbit Cube								
Corner	0	1	2	3	4	5	6	7
Stickers	(0, 1, 2)	(3, 4, 5)	(6, 7, 8)	(9, 10, 11)	(12, 13, 14)	(15, 16, 17)	(18, 19, 20)	(21, 22, 23)
Stickers	(D, O, F)	(A, E, I)	(B, L, S)	(C, R, P)	(W, J, H)	(X, G, N)	(U, M, Q)	(V, T, K)

If Corners 0, 1 and 2 are cycled in *that* order, then stickers (D, O, F), (A, E, I) and (B, L, S) are also cycled in the same order. There are 27 possible cases for cycling the 3 stickers belonging to the 3 different corner pieces. In the table below, it can be seen that sticker permutations (D A B), (O E L) and (F I S) are actually the same, because they give the same algorithm. The overall number of cases is then reduced from 27 to only 9.

Corners 3-cycle – Sticker Permutations & Algorithms – 9 Cases					
(D A B)	U2 R2 U' L' U R2 U' L U'	(O E L)	U2 R2 U' L' U R2 U' L U'	(F I S)	U2 R2 U' L' U R2 U' L U'
(D E B)	L2 D2 L U2 L' D2 L U2 L	(O I L)	L2 D2 L U2 L' D2 L U2 L	(F A S)	L2 D2 L U2 L' D2 L U2 L
(D I B)	D R2 D L2 D' R2 D L2 D2	(O A L)	D R2 D L2 D' R2 D L2 D2	(F E S)	D R2 D L2 D' R2 D L2 D2
(O A B)	U' R' D' R U R' D R	(F E L)	U' R' D' R U R' D R	(D I S)	U' R' D' R U R' D R
(O E B)	R' D' L D R D' L' D	(F I L)	R' D' L D R D' L' D	(D A S)	R' D' L D R D' L' D
(O I B)	R2 D2 L D R2 D' L' D R2 D R2	(F A L)	R2 D2 L D R2 D' L' D R2 D R2	(D E S)	R2 D2 L D R2 D' L' D R2 D R2
(F A B)	R' F2 R' B' R F2 R' B R2	(D E L)	R' F2 R' B' R F2 R' B R2	(O I S)	R' F2 R' B' R F2 R' B R2
(F E B)	U' F D F' U F D' F'	(D I L)	U' F D F' U F D' F'	(O A S)	U' F D F' U F D' F'
(F I B)	U2 B U' F2 U B' U' F2 U'	(D A L)	U2 B U' F2 U B' U' F2 U'	(O E S)	U2 B U' F2 U B' U' F2 U'
Corner Permutation: (Corner 0 → Corner 1 → Corner 2 → Corner 0)					

Taking corner twists into account, this means that:

1- there are only 9 different ways of cycling 3 corners for any given permutation of 3 pieces
2- sticker permutations can be used instead of {corner piece permutations + corner twists}

As there are only 8 corners, even permutations of corner pieces are limited to 3-, 5- and 7-cycles of corners. If 5- and 7-cycles are not available, they can still be built using only 3-cycles. Examples are given in the table below.

Corners 5-cycle & 7-cycle Built from 3-cycles – Examples
5-cycle – (D E S C J): F' D' B2 D F D' B2 D' L2 U L D2 L' U' L D2 L D2
7-cycle – (F E B P J X Q): U' B D' B' U B D' B U2 B' D2 B U2 B2 R' F L F' R F L' F'

Corner Orientation

This is to show how to extract corner twists from sticker vectors.

Let's consider sticker permutation (B R G) or (6 10 16): F2 R' F2 R' B2 R F2 R' B2 R2 F2

Reference sticker vector: (0, 1, 2, 3, 4, 5, **6**, **7**, **8**, **9**, **10**, **11**, 12, 13, 14, **15**, **16**, **17**, 18, 19, 20, 21, 22, 23)

Permuted sticker vector: (0, 1, 2, 3, 4, 5, **16**, **17**, **15**, **8**, **6**, **7**, 12, 13, 14, **9**, **10**, **11**, 18, 19, 20, 21, 22, 23)

Corner 0 permutation: (16 → 6, 17 → 7, 15 → 8)
Sticker group (16 17 15) must be shifted *right* to obtain a correct ordering of numbers, ie. in ascending order (15 16 17), thus Corner 0 twist is +1

Corner 1 permutation: (8 → 9, 6 → 10, 7 → 11)
Sticker group (8 6 7) must be shifted *left* to obtain a correct ordering of numbers, ie. in ascending order (6 7 8), thus Corner 1 twist is -1

Corner 2 permutation: (9 → 15, 10 → 16, 11 → 17)
Sticker group (9 10 11) need not be shifted because it is already in the right order, thus Corner 2 twist is 0

Conclusion:

1- if *right* shift, then corner twist = +1
2- if *left* shift, then corner twist = -1
3- if no shift, then corner twist = 0

Corner Parity

Corner parity is odd if corner pieces are permuted in such a way that:

1- the parity of the permutation of pieces is odd, or
2- the sum of corner twists, due to the permutation, is not equal to zero (Modulo 3)

The first condition is met if an *even* number of different pieces is cycled, ie for *single* 2-, 4-, 6- or 8-cycles of corners. In this case, there is permutation parity. In the second case there is an orientation parity. When there is corner parity, a few other pieces are also permuted, ie. edges and/or centers. Note that a corner permutation parity doesn't imply that there is automatically an orientation parity.

To check if there is corner parity, check if the number of permuted corners is even. If this is the case, corner parity should be fixed. To fix corner parity, simply do one ¼ turn of a single face or apply a corner swap algorithm. This will make the number of permuted corners odd, thus toggling corner parity from odd to even. We already know that if the number of permuted corner pieces is odd, then there is no permutation parity and no orientation parity and that any even permutation of corners can be solved with 3-cycles of stickers.

Corner Swap

A corner swap is a 2-cycle of corners where corners i and j are swapped. This can be used to fix corner parity. A corner swap algorithm may be composed of two parts. The first one is a dedge swap and the second is a T-Perm or N-Perm. Any of these 2 perms will swap 2 corners and 2 dedges. The second dedge swap will cancel out the first one, thus leaving a single corner swap plus a number of other permuted pieces.

Here is a corner swap example as applied to a 6x6x6 or 7x7x7 cube

1- Stefan's 5x5x5 (second) Dedge Swap applied to a 6x6x6 or 7x7x7 cube:
T3R2 T3F2 U2 VR2 U2 T3F2 T3R2 (7 moves)

2- T-Perm (to swap 2 *adjacent* corners and 2 opposed edges on a same face):
F' U' F U F R' F2 U F U F' U' F R (14 moves)

3- Modified T-Perm (to swap 2 *symmetrical* corners and 2 opposed edges on a same face):
R2 U F U' F' U' R U2 F' U' F' U F U' R (15 moves)

4- N-Perm (to swap 2 *opposed* corners on a same face and 2 opposed edges on a same face):
R U' L U2 R' U R L' U' L U2 R' U L' U (15 moves)

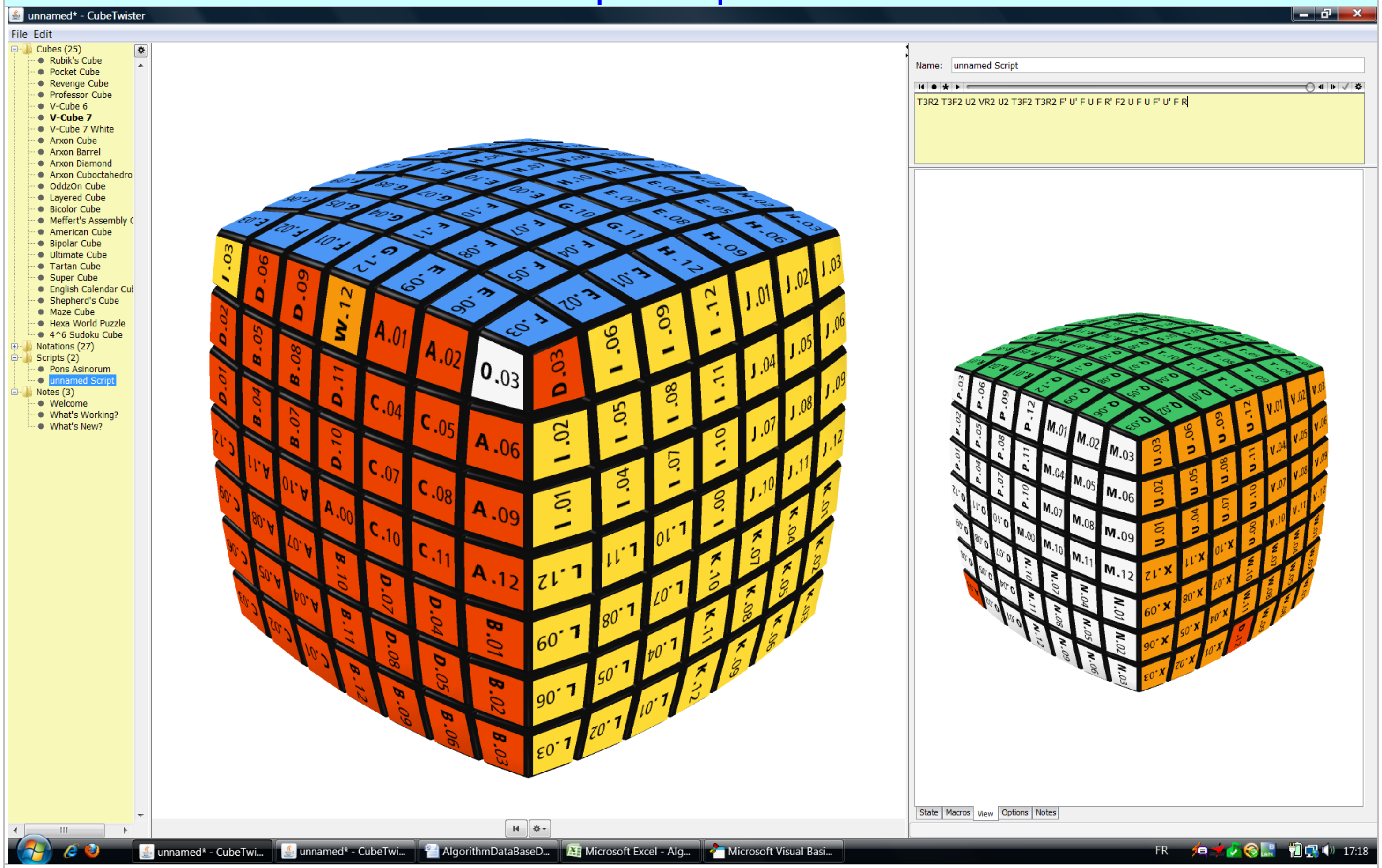
5- Dedge swap + T-Perm (to swap 2 adjacent corners):
T3R2 T3F2 U2 VR2 U2 T3F2 T3R2 F' U' F U F R' F2 U F U F' U' F R (21 moves)

6- Dedge swap + Modified T-Perm (to swap 2 symmetrical corners):
T3R2 T3U2 F2 VR2 F2 T3U2 T3R2 R2 U F U' F' U' R U2 F' U' F' U F U' R (22 moves)

7- Dedge swap + N-Perm (to swap 2 opposed corners on a same face):
T3R2 T3F2 U2 VR2 U2 T3F2 T3R2 R U' L U2 R' U R L' U' L U2 R' U L' U (22 moves)

From these example, templates can be built that will swap *any* 2 corners on a big cube. Note that for odd-order cubes, these algorithms will also swap 2 middle edges (midges are those edges located on a middle slice) and turn some centers on two faces, while preserving their color. This is shown on the picture below:

Corner Swap – Example – 7x7x7 Cube



Name: unnamed Script

T3R2 T3F2 U2 VR2 U2 T3F2 T3R2 F' U' F U F R' F2 U F U F' U' F R

Corner stickers A and D are swapped in Orbit 03 – Middle edge stickers A and W are also swapped (side effect)
T3R2 T3F2 U2 VR2 U2 T3F2 T3R2 F' U' F U F R' F2 U F U F' U' F R (21 moves)

Orbits and Scalable Algorithms

Corner D.03	Edge Outer Wing Left D.06 NF	Edge Inner Wing Left D.09 NF → N3F	Middle Edge Inner Wing D.12 MF	Edge Inner Wing Right A.01 NF → N3F	Edge Outer Wing Right A.02 NF	Corner A03
	Outer Corner-Center D.05 NF	Edge-Center Inner Wing Left D.08 NF N3F	Outer Middle-Center D.11 NF MF		Outer Corner-Center A.05 NF	
	Edge-Center Inner Wing Right D.04 NF → N3F N3F → NF	Inner Corner-Center D.07 NF → N3F	Inner Middle-Center D.10 NF → N3F MF	Inner Corner-Center A.07 NF → N3F		
			True Center A.00 F → T3F			

This 7x7 table shows how algorithms can be converted from one orbit to another. For example, if an algorithm exists for cycling Outer Wing Left Edges in orbit 06, then the same algorithm can be re-used for cycling Inner Wing Left Edges in orbit 09, simply by replacing 'N' with 'N3'. Similarly, if an algorithm exists for cycling Outer Corner-Centers in orbit 05, then the same algorithm can be re-used for cycling Inner Corner-Centers in orbit 07, simply by replacing 'N' with 'N3'. And finally, algorithms for orbits 04 and 08 can be exchanged simply by swapping 'N' and 'N3'.

Examples:
(I A E) Orbit 04 – 3-cycle of edge-centers (left) : F [NR, N3F R' N3F'] F'
(I A E) Orbit 08 – 3-cycle of edge-centers (right) : F [N3R, NF R' NF'] F'
(I A E) Orbit 11 – 3-cycle of midge-centers (outer) : F [MR, NF R' NF'] F'
(I A E) Orbit 10 – 3-cycle of midge-centers (inner) : F [MR, N3F R' N3F'] F'
(F A E) Orbit 05 – 3-cycle of corner-centers (outer) : F [NR, NF R' NF'] F'
(F A E) Orbit 07 – 3-cycle of corner-centers (inner) : F [N3R, N3F R' N3F'] F'
(I A O) Orbit 12 – 3-cycle of midges: F [MR, F R' F'] F'
(I A O) Orbit 06 – 3-cycle of edges: F [NR, F R' F'] F'

Part II

Algorithm Templates

Introduction

An algorithm template is a convenient way of representing a set of cube algorithms that will be executed on a computer. The shortest template is simply represented by letter X, which means any move from an *ordered* set of 12 outer layer quarter-turns: X = {F, F', R, R', U, U', L, L', D, D', B, B'}. Letter X' will then represent any outer layer quarter-turn from the associated *inverted* set: X' = {F', F, R', R, U', U, L', L, D', D, B', B} and letter X2 any outer layer half-turn from the square set: X2 = {F2, R2, U2, L2, D2, B2}. This holds for face moves only.

Similar templates also exist for slice moves, but moves in a set depend on the orbit in which the pieces are moving. For example, if we consider the first *edge* orbit, then MX will represent any slice quarter-turn from the set: MX = {NF, NF', NR, NR', NU, NU', NL, NL', ND, ND', NB, NB'}.

Slice quarter-turns for center orbits will be represented by different sets such as: MX = {NF, NF', NR, NR', NU, NU', NL, NL', ND, ND', NB, NB', N3F, N3F', N3R, N3R', N3U, N3U', N3L, N3L', N3D, N3D', N3B, N3B'}, for pieces that move in the first (NF) and second (N3F) slices, for example.

Complex templates can be built by concatenating basic templates such as X, X', X2, MX, MX' and MX2 and sweeping 'variables' of the set. By using nested *For…Next* loops to sweep variables, *all* occurences of a given template are then executed and each final cube state compared to the goal cube state. If there is a match between the two, then an algorithm has been found that will change the cube state from initial to goal. This will be better explained on an exemple. Let’s select template [(X Y X'), MZ], which is a Niklas commutator acting on edges:

[(X Y X'), MZ] = X Y X' MZ X Y' X' MZ'

There are three *independent* variables in this template, namely X, Y and MZ. Each variable is swept independently. This means that there are three nested *For…Next* loops for sweeping the 2 outer layer quarter-turns sets (X / X' and Y / Y') plus 1 slice quarter-turn set (MZ / MZ'). The search is exhaustive because *all* combinations represented in the template are checked. This is a brute force search method applied to finding algorithms.

Building Templates

Templates can be built after already existing algorithms. Consider, for example, the following algorithm:

(R U) (F' L F NR F' L' F NR') (U' R') (12-mover)

This cycles 3 edges (1 → 13 → 47 → 1) on face F. From this, we can build a more general template that could be used elsewhere on the cube, just by adding a *setup move*:

(X Y) [(Z P Z'), MQ] (X Y)' = X Y Z P Z' MQ Z P' Z' MQ' Y' X'

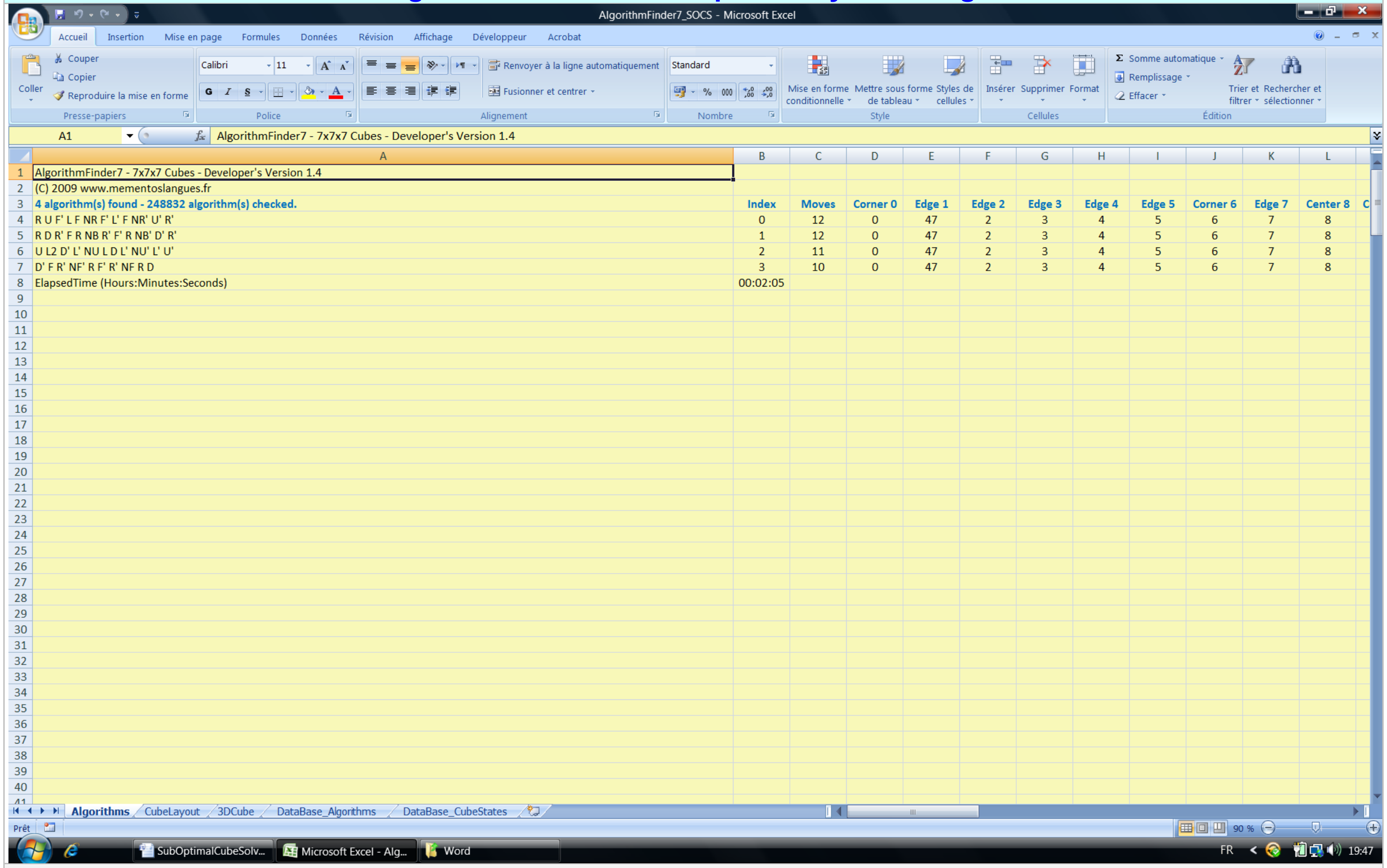
where R = X, U = Y, F' = Z, L = P and NR = MQ. This template of length 12 will then give a 3-cycle of edges at many places on the cube. By looking more closely at the template, we can see that this is also a Niklas commutator [(Z P Z'), MQ] combined with a particular setup move (X Y).

Sometimes, algorithms *shorter* than the template may even be found, thanks to *cancellations* or *combinations* of a few consecutive moves. In the example above, we can see that Y and Z do cancel each other out if Y = Z'. A shorter 10-move algorithm, which cycles exactly the same 3 edges as the previous 12-mover, can then be used, where Y = Z' = R':

D' F R' NF' R F' R' NF R D (10-mover)

Finding the shortest (fewest move) algorithm when executing a template does not mean that this is an optimal one in terms of 'absolute' number of moves. It only means that no shorter algorithm will be found using *this* particular template. This is why it is better to use a set of different templates until the shortest algorithm among a given set of templates has been found.

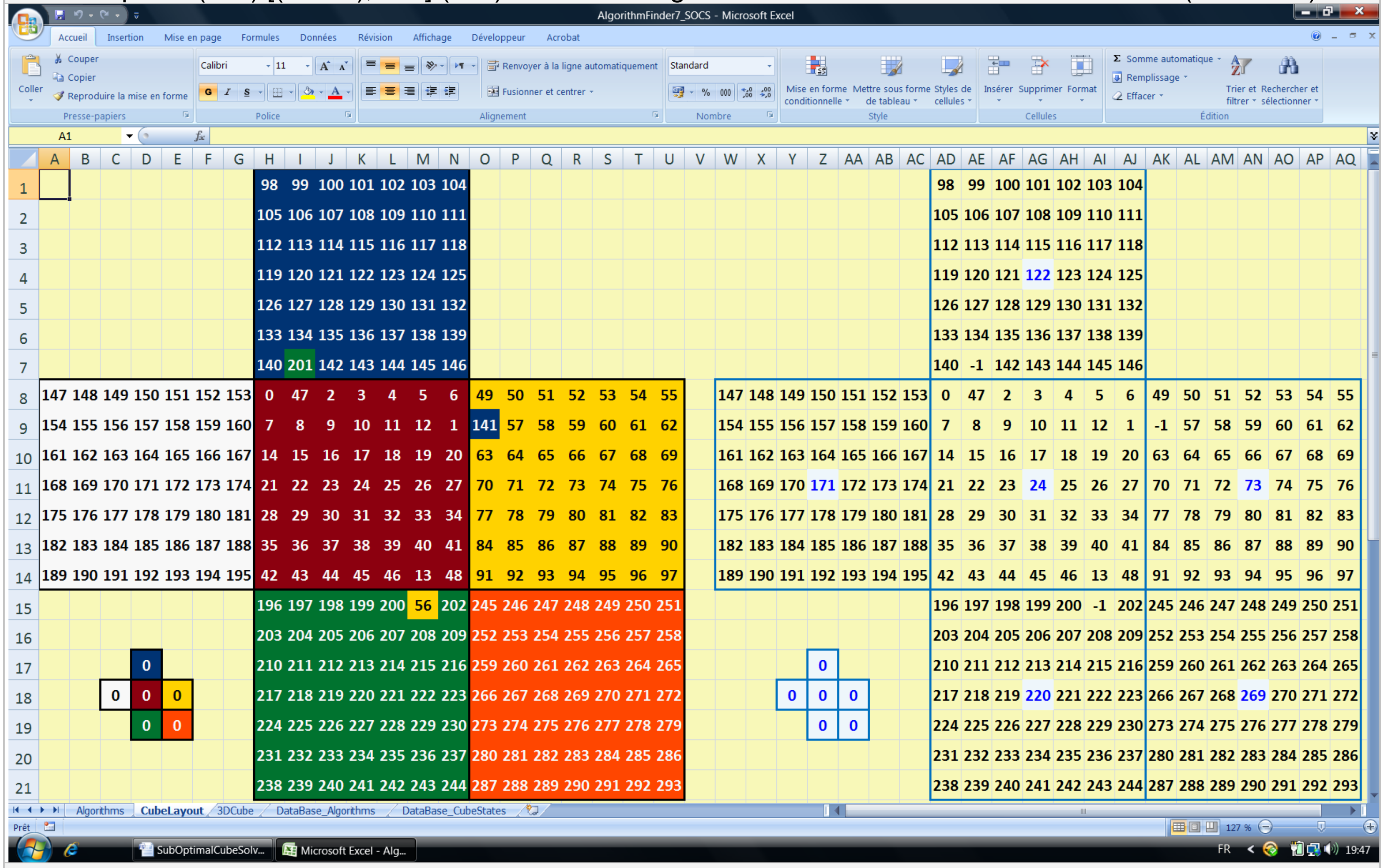
AlgorithmFinder7 – Example – 3-Cycle of Edges



A1: AlgorithmFinder7 - 7x7x7 Cubes - Developer's Version 1.4

	A	B	C	D	E	F	G	H	I	J	K	L
1	AlgorithmFinder7 - 7x7x7 Cubes - Developer's Version 1.4											
2	(C) 2009 www.mementoslangues.fr											
3	4 algorithm(s) found - 248832 algorithm(s) checked.	Index	Moves	Corner 0	Edge 1	Edge 2	Edge 3	Edge 4	Edge 5	Corner 6	Edge 7	Center 8
4	R U F' L F NR F' L' F NR' U' R'	0	12	0	47	2	3	4	5	6	7	8
5	R D R' F R NB R' F' R NB' D' R'	1	12	0	47	2	3	4	5	6	7	8
6	U L2 D' L' NU L D L' NU' L' U'	2	11	0	47	2	3	4	5	6	7	8
7	D' F R' NF' R F' R' NF R D	3	10	0	47	2	3	4	5	6	7	8
8	ElapsedTime (Hours:Minutes:Seconds)	00:02:05										

Template: (X Y) [(Z P Z'), MQ] (X Y)' – Shortest algorithm found: D' F R' NF' R F' R' NF R D (10-mover)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ
1								98	99	100	101	102	103	104																98	99	100	101	102	103	104							
2								105	106	107	108	109	110	111																105	106	107	108	109	110	111							
3								112	113	114	115	116	117	118																112	113	114	115	116	117	118							
4								119	120	121	122	123	124	125																119	120	121	122	123	124	125							
5								126	127	128	129	130	131	132																126	127	128	129	130	131	132							
6								133	134	135	136	137	138	139																133	134	135	136	137	138	139							
7								140	201	142	143	144	145	146																140	-1	142	143	144	145	146							
8	147	148	149	150	151	152	153	0	47	2	3	4	5	6	49	50	51	52	53	54	55		147	148	149	150	151	152	153	0	47	2	3	4	5	6	49	50	51	52	53	54	55
9	154	155	156	157	158	159	160	7	8	9	10	11	12	1	141	57	58	59	60	61	62		154	155	156	157	158	159	160	7	8	9	10	11	12	1	-1	57	58	59	60	61	62
10	161	162	163	164	165	166	167	14	15	16	17	18	19	20	63	64	65	66	67	68	69		161	162	163	164	165	166	167	14	15	16	17	18	19	20	63	64	65	66	67	68	69
11	168	169	170	171	172	173	174	21	22	23	24	25	26	27	70	71	72	73	74	75	76		168	169	170	171	172	173	174	21	22	23	24	25	26	27	70	71	72	73	74	75	76
12	175	176	177	178	179	180	181	28	29	30	31	32	33	34	77	78	79	80	81	82	83		175	176	177	178	179	180	181	28	29	30	31	32	33	34	77	78	79	80	81	82	83
13	182	183	184	185	186	187	188	35	36	37	38	39	40	41	84	85	86	87	88	89	90		182	183	184	185	186	187	188	35	36	37	38	39	40	41	84	85	86	87	88	89	90
14	189	190	191	192	193	194	195	42	43	44	45	46	13	48	91	92	93	94	95	96	97		189	190	191	192	193	194	195	42	43	44	45	46	13	48	91	92	93	94	95	96	97
15								196	197	198	199	200	56	202	245	246	247	248	249	250	251									196	197	198	199	200	-1	202	245	246	247	248	249	250	251
16								203	204	205	206	207	208	209	252	253	254	255	256	257	258									203	204	205	206	207	208	209	252	253	254	255	256	257	258
17				0				210	211	212	213	214	215	216	259	260	261	262	263	264	265					0				210	211	212	213	214	215	216	259	260	261	262	263	264	265
18			0	0	0			217	218	219	220	221	222	223	266	267	268	269	270	271	272				0	0	0			217	218	219	220	221	222	223	266	267	268	269	270	271	272
19				0	0			224	225	226	227	228	229	230	273	274	275	276	277	278	279					0	0			224	225	226	227	228	229	230	273	274	275	276	277	278	279
20								231	232	233	234	235	236	237	280	281	282	283	284	285	286									231	232	233	234	235	236	237	280	281	282	283	284	285	286
21								238	239	240	241	242	243	244	287	288	289	290	291	292	293									238	239	240	241	242	243	244	287	288	289	290	291	292	293

3-cycle of edges on face F (1 → 13 → 47 →1)

Shifted Templates

When a template has been found for cycling some pieces, other templates for cycling other pieces can also be found, simply by *shifting* variables inside a template, as shown in the example below:

A [B, MX] A' = A B MX B' MX' A'

If this template is shifted, a reduction in number of moves may be obtained just by cancelling 2 consecutive moves:

A B MX B' MX' **A'** → **A' A** B MX B' MX' = B MX B' MX'

The end result is a template that will also cycle similar pieces on the cube. Here is an example:

D' F R' NF' R F' R' NF R **D** (10-mover)

D' F R' NF' R F' R' NF R **D** → **D D'** F R' NF' R F' R' NF R = F R' NF' R F' R' NF R

F R' NF' R F' R' NF **R** → **R F** R' NF' R F' R' **NF** → **NF R F** R' NF' R F' **R'** → **R' NF R F** R' NF' R F' etc…

Inverted Templates

If a template is used to find an algorithm for a 3-cycle of stickers, the same but inverted (or primed) template will also find an algorithm for another 3-cycle of the *same* stickers, but cycled in a different order, where 2 out of 3 stickers are simply swapped. Let's consider a 3-cycle of centers on a 7x7x7 cube: [NL, NU' R' NU]. This Niklas commutator will cycle 3 centers, whereas the inverted commutator [NL, NU' R' NU]' = [NU' R' NU, NL] will cycle the same stickers, but in a different order:

3-cycle Alg: (8 → 12 → 106 → 8)
3-cycle Alg': (8 → 106 → 12 → 8)

This shows that templates for 3-cycles of stickers should come in pairs:

a pair of templates = template + inverted template

Niklas Commutators

Niklas commutators are very useful for building 3-cycles of corners, edges or centers, without modifying other pieces on the cube. There are many possible variations, depending of the type of pieces to be cycled and on the outer layers and slices that are involved. The basic Niklas commutator is an 8-mover. By adding appropriate setup moves, triplets of stickers in a same orbit can be cycled. Setup moves of 0, 1, 2 or 3 moves are sufficient for cycling *any* triplet. This means that *any* triplet of stickers in a same orbit can be cycled simply by using an optimal combination of {setup move + Niklas commutator}.

Templates – Niklas Commutators – Centers 3-Cycle	
Commutator	**Inverted Commutator**
[MX, Y MZ Y']	[X MY X', MZ]
[X, MY MZ MY']	[MX MY MX', Z]
[MX, MY Z MY']	[MX Y MX', MZ]
Templates – Commutators + Setup Moves – Centers 3-Cycle	
Commutator + Setup Move	**Inverted Commutator + Setup Move**
(X) [ ] (X)'	(X) [ ]' (X)'
(MX) [ ] (MX)'	(MX) [ ]' (MX)'
(X MY) [ ] (X MY)'	(X MY) [ ]' (X MY)'
(MX Y) [ ] (MX Y)'	(MX Y) [ ]' (MX Y)'
(X MY MZ) [ ] (X MY MZ)'	(X MY MZ) [ ]' (X MY MZ)'
(MX Y MZ) [ ] (MX Y MZ)'	(MX Y MZ) [ ]' (MX Y MZ)'
(MX MY Z) [ ] (MX MY Z)'	(MX MY Z) [ ]' (MX MY Z)'
(X Y MZ) [ ] (X Y MZ)'	(X Y MZ) [ ]' (X Y MZ)'
(X MZ Y) [ ] (X MZ Y)'	(X MZ Y) [ ]' (X MZ Y)'
(MX Y Z) [ ] (MX Y Z)'	(MX Y Z) [ ]' (MX Y Z)'

Generic & Specific Templates

Generic templates are general purpose templates which are applicable to a wide range of cases. They usually use a maximum number of variables and are therefore slow to execute. Specific templates are derived from generic templates by introducing dependencies between a few variables so as to decrease computing time. It's a good practice not to use more than 4 variables in a specific template.

Generic & Specific Templates Examples – Niklas Commutators – Centers 3-Cycle			
Generic	**Variables**	**Specific**	**Variables**
(X MY) [Z, MP MQ MP'] (X MY)'	5	(X MY) [X', MZ MP MZ'] (X MY)'	4
(MX MY Z) [MP, MQ V MQ'] (MX MY Z)'	6	(MX MY Y') [MZ, MP Z' MP'] (MX MY Y')'	4
(MX Y MZ) [P, MQ V MQ'] (MX Y MZ)'	6	(MX Y MX) [Z, MP Z' MP'] (MX Y MX)'	4
(X Y MZ) [P, MQ MV MQ'] (X Y MZ)'	6	(X Y MZ) [Y, MP MY MP'] (X Y MZ)'	4

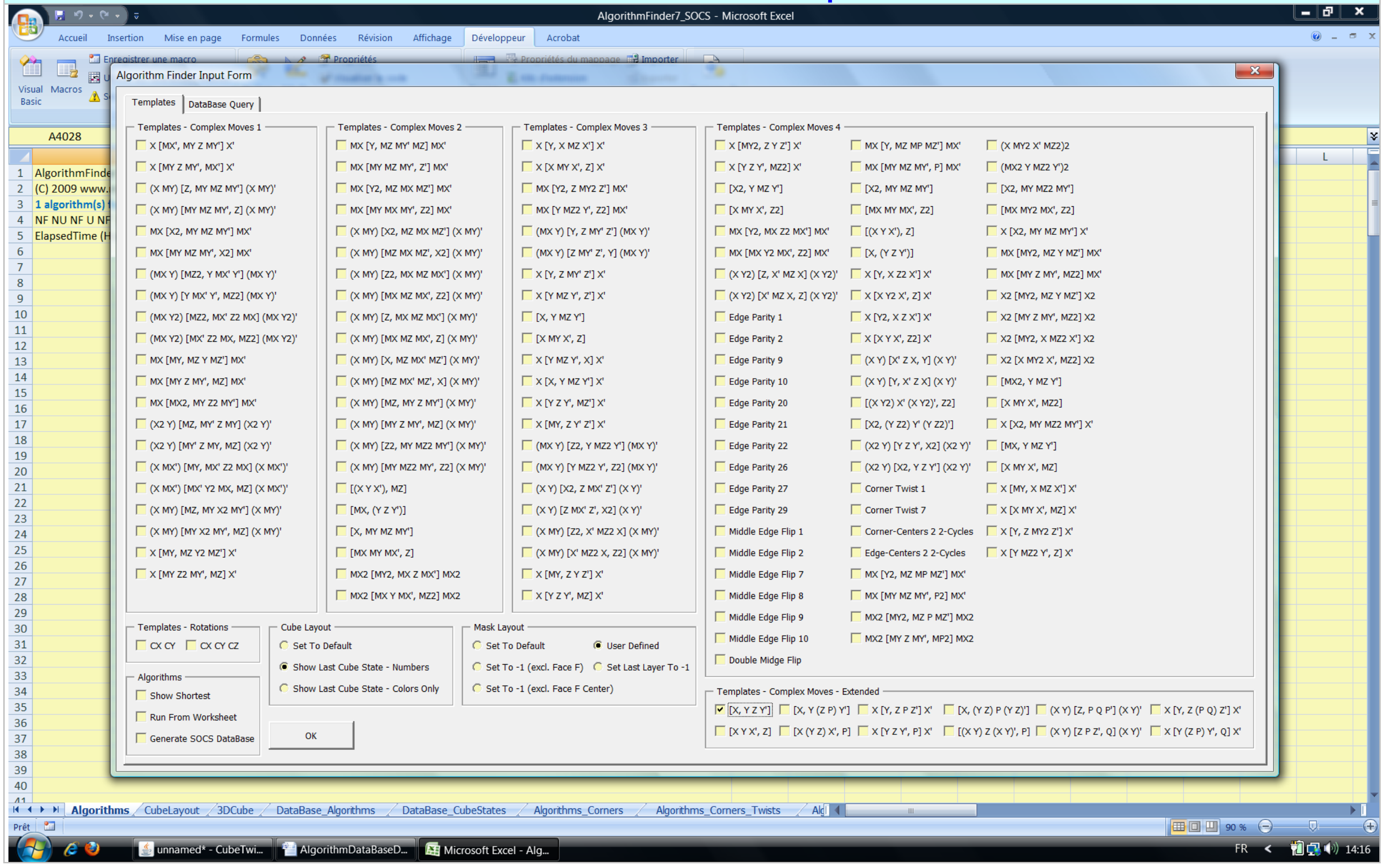
Extended Templates

Extended templates are special templates which are applicable to 'difficult to solve' cases. They use *all* available variables, including quarter-turn and half-turn variables and are therefore very slow to execute.

Extended Templates – Niklas Commutators			
Commutator	**Variables**	**Inverted Commutator**	**Variables**
[X, Y Z Y']	3	[X Y X', Z]	3
[X, Y (Z P) Y']	4	[X (Y Z) X', P]	4
X [Y, Z P Z'] X'	4	X [Y Z Y', P] X'	4
[X, (Y Z) P (Y Z)']	4	[(X Y) Z (X Y)', P]	4
(X Y) [Z, P Q P'] (X Y)'	5	(X Y) [Z P Z', Q] (X Y)'	5
X [Y, Z (P Q) Z'] X'	5	X [Y (Z P) Y', Q] X'	5

Below is an example of an extended set of moves with up to 55 variables:
X = {F2, R2, U2, L2, D2, B2, NF2, NR2, NU2, NL2, ND2, NB2, N3F2, N3R2, N3U2, N3L2, N3D2, N3B2, F, Fi, R, Ri, U, Ui, L, Li, D, Di, B, Bi, NF, NFi, NR, NRi, NU, NUi, NL, NLi, ND, NDi, NB, NBi, N3F, N3Fi, N3R, N3Ri, N3U, N3Ui, N3L, N3Li, N3D, N3Di, N3B, N3Bi, NoMove}

UserForm – Extended Templates



Algorithm Finder Input Form

Templates | DataBase Query

Templates - Complex Moves 1
- ☐ X [MX', MY Z MY'] X'
- ☐ X [MY Z MY', MX'] X'
- ☐ (X MY) [Z, MY MZ MY'] (X MY)'
- ☐ (X MY) [MY MZ MY', Z] (X MY)'
- ☐ MX [X2, MY MZ MY'] MX'
- ☐ MX [MY MZ MY', X2] MX'
- ☐ (MX Y) [MZ2, Y MX' Y'] (MX Y)'
- ☐ (MX Y) [Y MX' Y', MZ2] (MX Y)'
- ☐ (MX Y2) [MZ2, MX' Z2 MX] (MX Y2)'
- ☐ (MX Y2) [MX' Z2 MX, MZ2] (MX Y2)'
- ☐ MX [MY, MZ Y MZ'] MX'
- ☐ MX [MY Z MY', MZ] MX'
- ☐ MX [MX2, MY Z2 MY'] MX'
- ☐ (X2 Y) [MZ, MY' Z MY] (X2 Y)'
- ☐ (X2 Y) [MY' Z MY, MZ] (X2 Y)'
- ☐ (X MX') [MY, MX' Z2 MX] (X MX')'
- ☐ (X MX') [MX' Y2 MX, MZ] (X MX')'
- ☐ (X MY) [MZ, MY X2 MY'] (X MY)'
- ☐ (X MY) [MY X2 MY', MZ] (X MY)'
- ☐ X [MY, MZ Y2 MZ'] X'
- ☐ X [MY Z2 MY', MZ] X'

Templates - Complex Moves 2
- ☐ MX [Y, MZ MY' MZ] MX'
- ☐ MX [MY MZ MY', Z'] MX'
- ☐ MX [Y2, MZ MX MZ'] MX'
- ☐ MX [MY MX MY', Z2] MX'
- ☐ (X MY) [X2, MZ MX MZ'] (X MY)'
- ☐ (X MY) [MZ MX MZ', X2] (X MY)'
- ☐ (X MY) [Z2, MX MZ MX'] (X MY)'
- ☐ (X MY) [MX MZ MX', Z2] (X MY)'
- ☐ (X MY) [Z, MX MZ MX'] (X MY)'
- ☐ (X MY) [MX MZ MX', Z] (X MY)'
- ☐ (X MY) [X, MZ MX' MZ'] (X MY)'
- ☐ (X MY) [MZ MX' MZ', X] (X MY)'
- ☐ (X MY) [MZ, MY Z MY'] (X MY)'
- ☐ (X MY) [MY Z MY', MZ] (X MY)'
- ☐ (X MY) [Z2, MY MZ2 MY'] (X MY)'
- ☐ (X MY) [MY MZ2 MY', Z2] (X MY)'
- ☐ [(X Y X'), MZ]
- ☐ [MX, (Y Z Y')]
- ☐ [X, MY MZ MY']
- ☐ [MX MY MX', Z]
- ☐ MX2 [MY2, MX Z MX'] MX2
- ☐ MX2 [MX Y MX', MZ2] MX2

Templates - Complex Moves 3
- ☐ X [Y, X MZ X'] X'
- ☐ X [X MY X', Z] X'
- ☐ MX [Y2, Z MY2 Z'] MX'
- ☐ MX [Y MZ2 Y', Z2] MX'
- ☐ (MX Y) [Y, Z MY' Z'] (MX Y)'
- ☐ (MX Y) [Z MY' Z', Y] (MX Y)'
- ☐ X [Y, Z MY' Z'] X'
- ☐ X [Y MZ Y', Z'] X'
- ☐ [X, Y MZ Y']
- ☐ [X MY X', Z]
- ☐ X [Y MZ Y', X] X'
- ☐ X [X, Y MZ Y'] X'
- ☐ X [Y Z Y', MZ'] X'
- ☐ X [MY, Z Y' Z'] X'
- ☐ (MX Y) [Z2, Y MZ2 Y'] (MX Y)'
- ☐ (MX Y) [Y MZ2 Y', Z2] (MX Y)'
- ☐ (X Y) [X2, Z MX' Z'] (X Y)'
- ☐ (X Y) [Z MX' Z', X2] (X Y)'
- ☐ (X MY) [Z2, X' MZ2 X] (X MY)'
- ☐ (X MY) [X' MZ2 X, Z2] (X MY)'
- ☐ X [MY, Z Y Z'] X'
- ☐ X [Y Z Y', MZ] X'

Templates - Complex Moves 4

☐ X [MY2, Z Y Z'] X'	☐ MX [Y, MZ MP MZ'] MX'	☐ (X MY2 X' MZ2)2
☐ X [Y Z Y', MZ2] X'	☐ MX [MY MZ MY', P] MX'	☐ (MX2 Y MZ2 Y')2
☐ [X2, Y MZ Y']	☐ [X2, MY MZ MY']	☐ [X2, MY MZ2 MY']
☐ [X MY X', Z2]	☐ [MX MY MX', Z2]	☐ [MX MY2 MX', Z2]
☐ MX [Y2, MX Z2 MX'] MX'	☐ [(X Y X'), Z]	☐ X [X2, MY MZ MY'] X'
☐ MX [MX Y2 MX', Z2] MX'	☐ [X, (Y Z Y')]	☐ MX [MY2, MZ Y MZ'] MX'
☐ (X Y2) [Z, X' MZ X] (X Y2)'	☐ X [Y, X Z2 X'] X'	☐ MX [MY Z MY', MZ2] MX'
☐ (X Y2) [X' MZ X, Z] (X Y2)'	☐ X [X Y2 X', Z] X'	☐ X2 [MY2, MZ Y MZ'] X2
☐ Edge Parity 1	☐ X [Y2, X Z X'] X'	☐ X2 [MY Z MY', MZ2] X2
☐ Edge Parity 2	☐ X [X Y X', Z2] X'	☐ X2 [MY2, X MZ2 X'] X2
☐ Edge Parity 9	☐ (X Y) [X' Z X, Y] (X Y)'	☐ X2 [X MY2 X', MZ2] X2
☐ Edge Parity 10	☐ (X Y) [Y, X' Z X] (X Y)'	☐ [MX2, Y MZ Y']
☐ Edge Parity 20	☐ [(X Y2) X' (X Y2)', Z2]	☐ [X MY X', MZ2]
☐ Edge Parity 21	☐ [X2, (Y Z2) Y' (Y Z2)']	☐ X [X2, MY MZ2 MY'] X'
☐ Edge Parity 22	☐ (X2 Y) [Y Z Y', X2] (X2 Y)'	☐ [MX, Y MZ Y']
☐ Edge Parity 26	☐ (X2 Y) [X2, Y Z Y'] (X2 Y)'	☐ [X MY X', MZ]
☐ Edge Parity 27	☐ Corner Twist 1	☐ X [MY, X MZ X'] X'
☐ Edge Parity 29	☐ Corner Twist 7	☐ X [X MY X', MZ] X'
☐ Middle Edge Flip 1	☐ Corner-Centers 2 2-Cycles	☐ X [Y, Z MY2 Z'] X'
☐ Middle Edge Flip 2	☐ Edge-Centers 2 2-Cycles	☐ X [Y MZ2 Y', Z] X'
☐ Middle Edge Flip 7	☐ MX [Y2, MZ MP MZ'] MX'	
☐ Middle Edge Flip 8	☐ MX [MY MZ MY', P2] MX'	
☐ Middle Edge Flip 9	☐ MX2 [MY2, MZ P MZ'] MX2	
☐ Middle Edge Flip 10	☐ MX2 [MY Z MY', MP2] MX2	
☐ Double Midge Flip		

Templates - Rotations
- ☐ CX CY ☐ CX CY CZ

Algorithms
- ☐ Show Shortest
- ☐ Run From Worksheet
- ☐ Generate SOCS DataBase

Cube Layout
- ○ Set To Default
- ● Show Last Cube State - Numbers
- ○ Show Last Cube State - Colors Only

Mask Layout
- ○ Set To Default ● User Defined
- ○ Set To -1 (excl. Face F) ○ Set Last Layer To -1
- ○ Set To -1 (excl. Face F Center)

OK

Templates - Complex Moves - Extended

✓ [X, Y Z Y']	☐ [X, Y (Z P) Y']	☐ X [Y, Z P Z'] X'	☐ [X, (Y Z) P (Y Z)']	☐ (X Y) [Z, P Q P'] (X Y)'	☐ X [Y, Z (P Q) Z'] X'
☐ [X Y X', Z]	☐ [X (Y Z) X', P]	☐ X [Y Z Y', P] X'	☐ [(X Y) Z (X Y)', P]	☐ (X Y) [Z P Z', Q] (X Y)'	☐ X [Y (Z P) Y', Q] X'

Scalable Algorithms

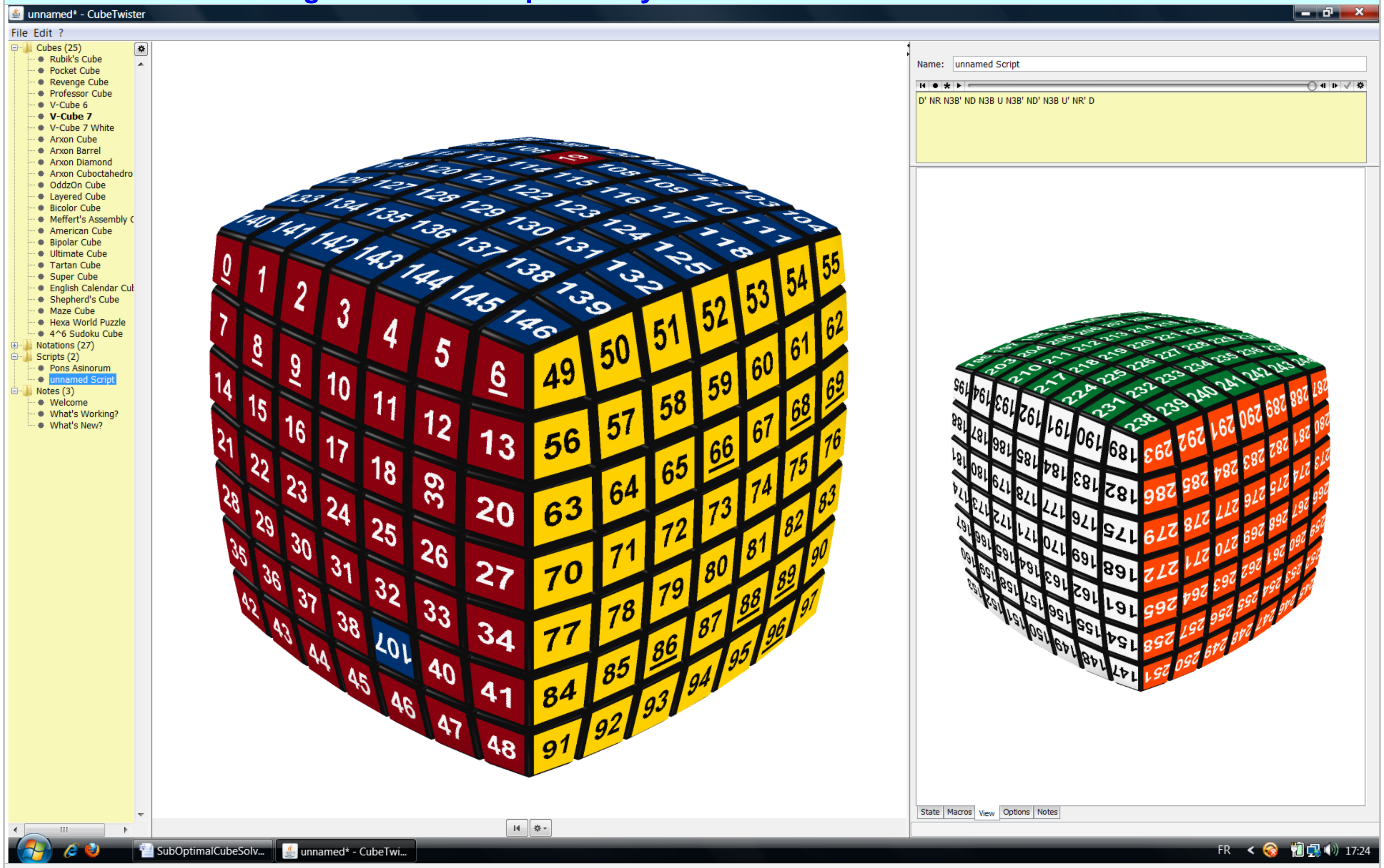
When running a template, the shortest algorithms found for 3-cycles of stickers are selected if and only if the computed final cube state and the goal state are identical. The selection criteria specifies that all other stickers shall stay in place. In other words, any algorithm found will modify the locations of only 3 stickers in the same orbit. This is a very useful feature because now these algorithms are perfectly scalable to any cube order N and can also be re-used for 3-cycles of stickers on other slices. This is best shown on an example. Let’s consider a 3-cycle of centers on a 7x7x7 cube:

D' NR NB' ND NB U NB' ND' NB U' NR' D = (D' NR) [NB' ND NB, U] (D' NR)'

This algorithm will also cycle 3 center stickers on the *same* locations on a 17x17x17 cube. Moreover, by replacing slice move NB with N3B, a 3-cycle of centers in another orbit is obtained:

D' NR N3B' ND N3B U N3B' ND' N3B U' NR' D = (D' NR) [N3B' ND N3B, U] (D' NR)'

Scalable Algorithms – Example – 3-Cycle of Centers – Extension To Another Orbit

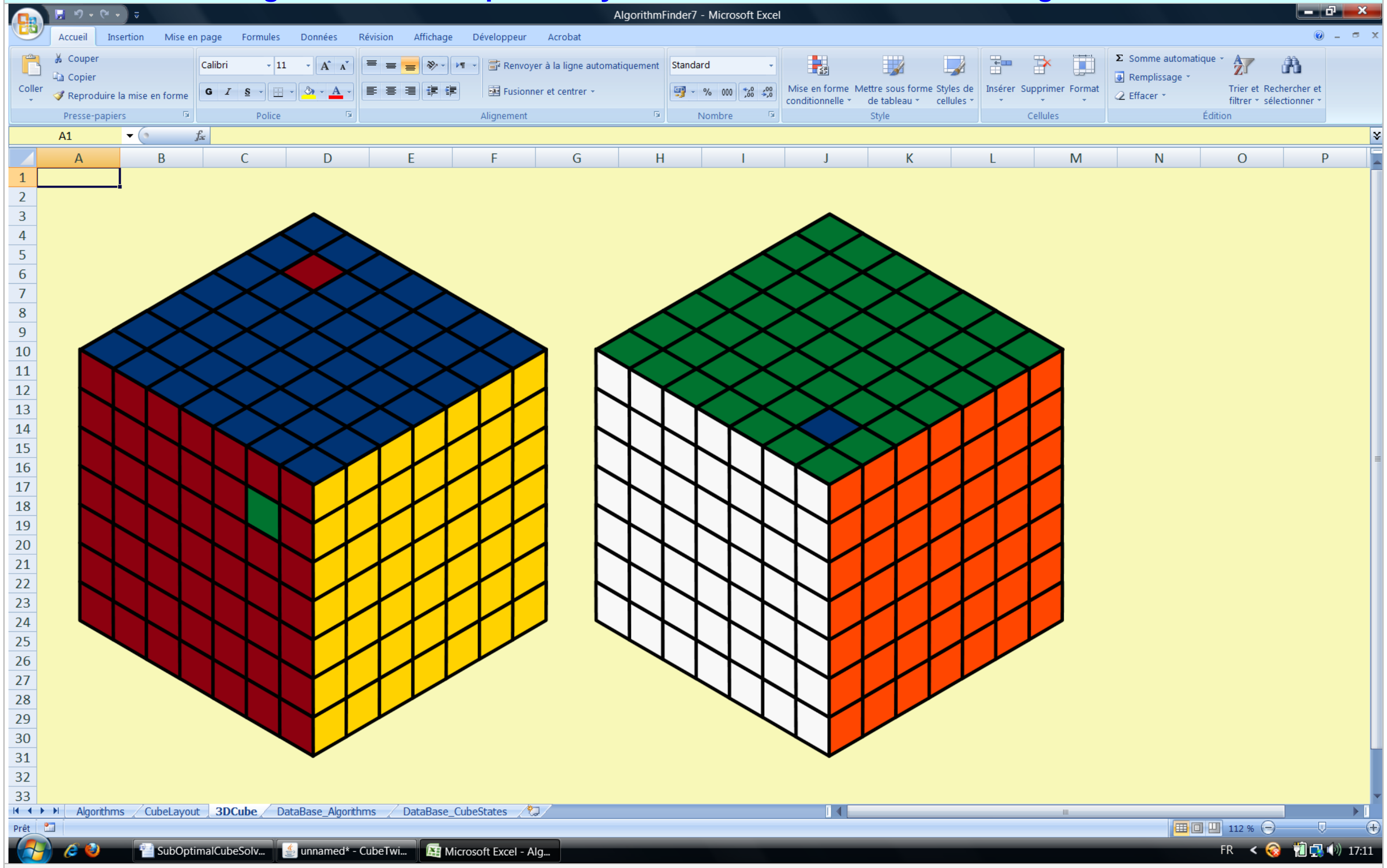


Name: unnamed Script

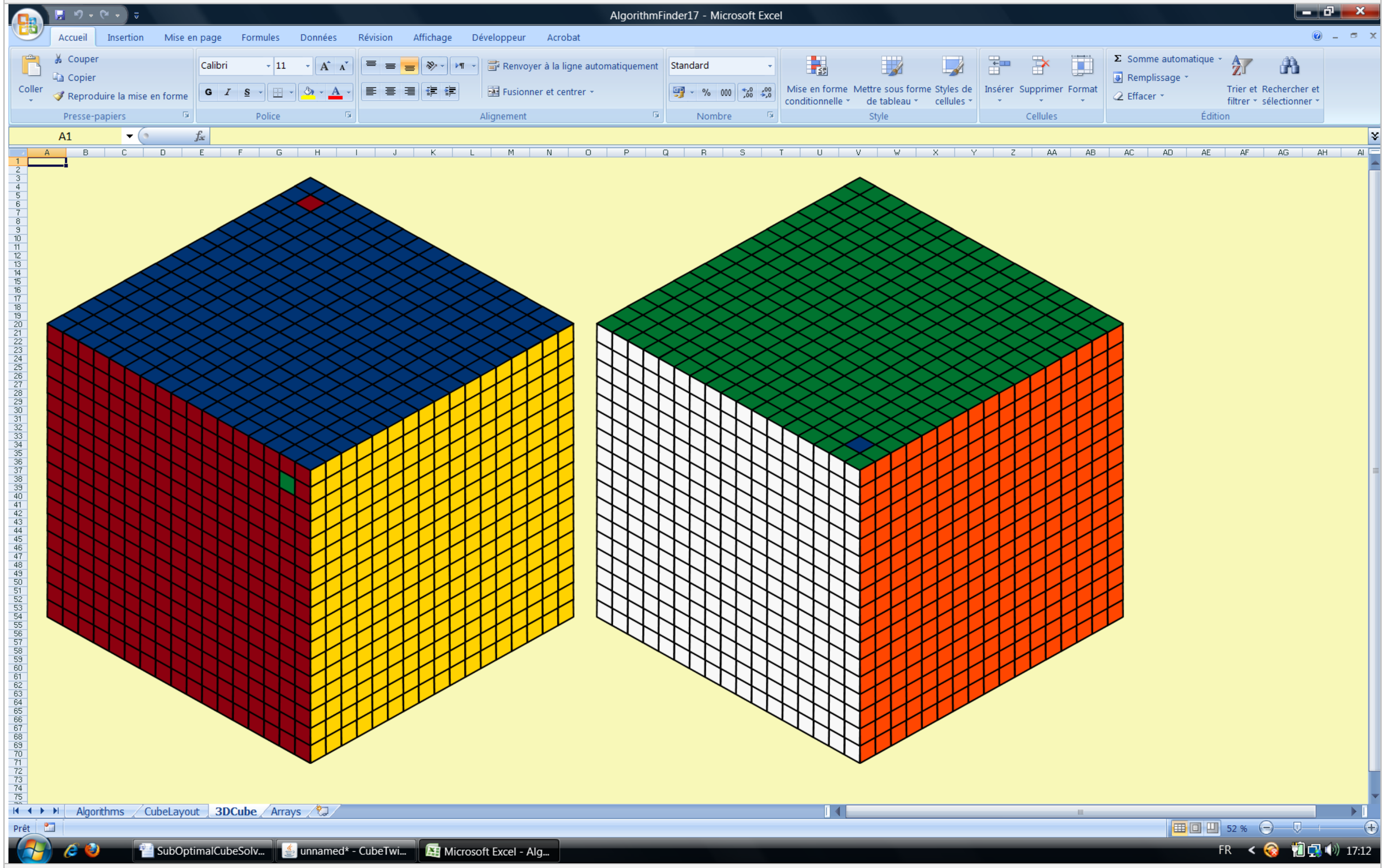
D' NR N3B' ND N3B U N3B' ND' N3B U' NR' D

7x7x7 Cube – D' NR N3B' ND N3B U N3B' ND' N3B U' NR' D – (19 → 107 → 39 → 19)

Scalable Algorithms – Example – 3-Cycle of Centers – Extension To Higher Order Cubes

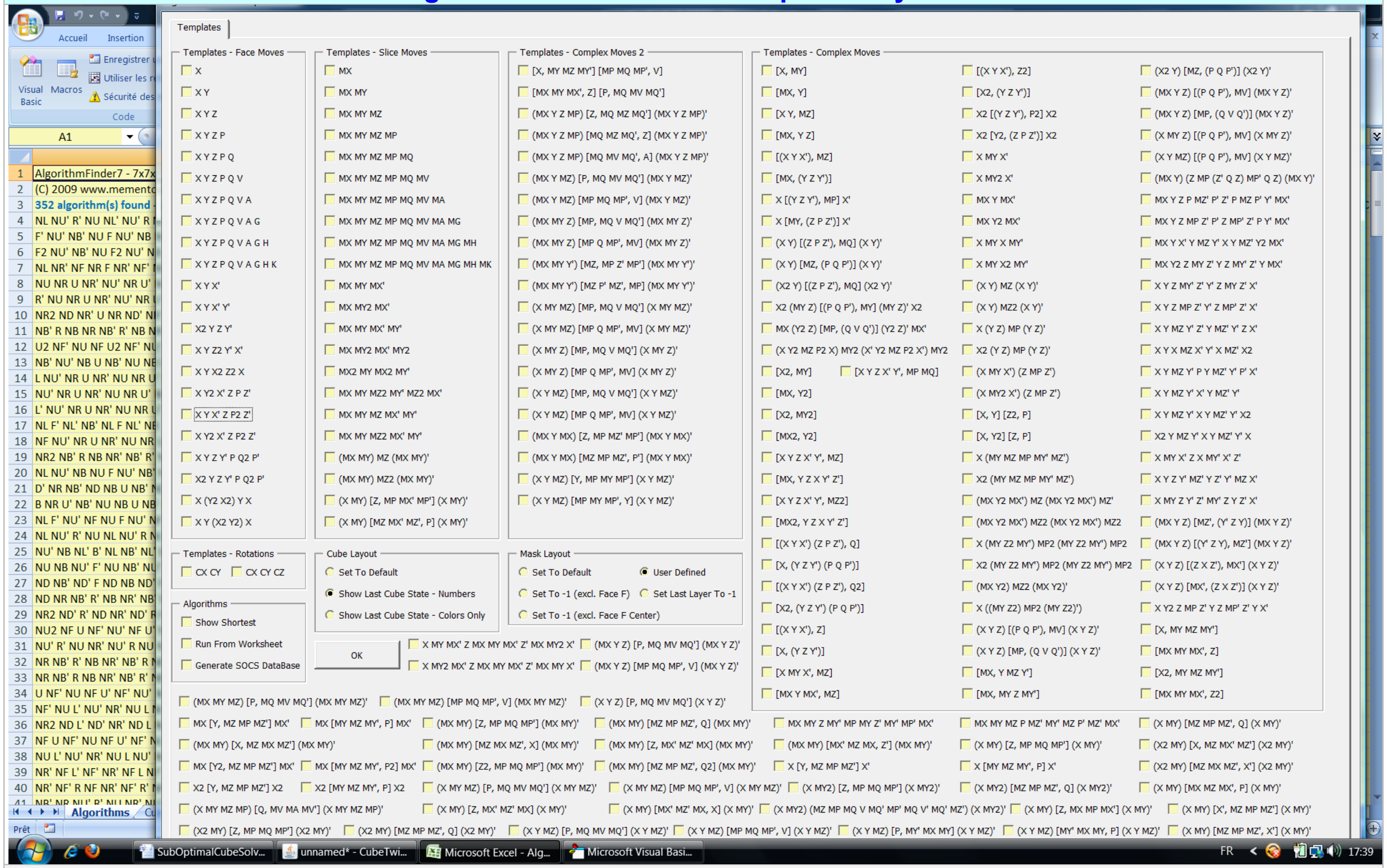


7x7x7 Cube – D' NR NB' ND NB U NB' ND' NB U' NR' D



17x17x17 Cube – D' NR NB' ND NB U NB' ND' NB U' NR' D

SOCS7 Algorithms DataBase – Example – 3-Cycle of Centers



Templates

Templates - Face Moves
- X
- X Y
- X Y Z
- X Y Z P
- X Y Z P Q
- X Y Z P Q V
- X Y Z P Q V A
- X Y Z P Q V A G
- X Y Z P Q V A G H
- X Y Z P Q V A G H K
- X Y X'
- X Y X' Y'
- X2 Y Z Y'
- X Y Z2 Y' X'
- X Y X2 Z2 X
- X Y2 X' Z P Z'
- X Y X' Z P2 Z'
- X Y2 X' Z P2 Z'
- X Y Z Y' P Q2 P'
- X2 Y Z Y' P Q2 P'
- X (Y2 X2) Y X
- X Y (X2 Y2) X

Templates - Slice Moves
- MX
- MX MY
- MX MY MZ
- MX MY MZ MP
- MX MY MZ MP MQ
- MX MY MZ MP MQ MV
- MX MY MZ MP MQ MV MA
- MX MY MZ MP MQ MV MA MG
- MX MY MZ MP MQ MV MA MG MH
- MX MY MZ MP MQ MV MA MG MH MK
- MX MY MX'
- MX MY2 MX'
- MX MY MX' MY'
- MX MY2 MX' MY2
- MX2 MY MX2 MY'
- MX MY MZ2 MY' MZ2 MX'
- MX MY MZ MX' MY'
- MX MY MZ2 MX' MY'
- (MX MY) MZ (MX MY)'
- (MX MY) MZ2 (MX MY)'
- (X MY) [Z, MP MX' MP'] (X MY)'
- (X MY) [MZ MX' MZ', P] (X MY)'

Templates - Complex Moves 2
- [X, MY MZ MY'] [MP MQ MP', V]
- [MX MY MX', Z] [P, MQ MV MQ']
- (MX Y Z MP) [Z, MQ MZ MQ'] (MX Y Z MP)'
- (MX Y Z MP) [MQ MZ MQ', Z] (MX Y Z MP)'
- (MX Y Z MP) [MQ MV MQ', A] (MX Y Z MP)'
- (MX Y MZ) [P, MQ MV MQ'] (MX Y MZ)'
- (MX Y MZ) [MP MQ MP', V] (MX Y MZ)'
- (MX MY Z) [MP, MQ V MQ'] (MX MY Z)'
- (MX MY Z) [MP Q MP', MV] (MX MY Z)'
- (MX MY Y') [MZ, MP Z' MP'] (MX MY Y')'
- (MX MY Y') [MZ P' MZ', MP] (MX MY Y')'
- (X MY MZ) [MP, MQ V MQ'] (X MY MZ)'
- (X MY MZ) [MP Q MP', MV] (X MY MZ)'
- (X MY Z) [MP, MQ V MQ'] (X MY Z)'
- (X MY Z) [MP Q MP', MV] (X MY Z)'
- (X Y MZ) [MP, MQ V MQ'] (X Y MZ)'
- (X Y MZ) [MP Q MP', MV] (X Y MZ)'
- (MX Y MX) [Z, MP MZ' MP'] (MX Y MX)'
- (MX Y MX) [MZ MP MZ', P'] (MX Y MX)'
- (X Y MZ) [Y, MP MY MP'] (X Y MZ)'
- (X Y MZ) [MP MY MP', Y] (X Y MZ)'

Templates - Complex Moves
- [X, MY]
- [MX, Y]
- [X Y, MZ]
- [MX, Y Z]
- [(X Y X'), MZ]
- [MX, (Y Z Y')]
- X [(Y Z Y'), MP] X'
- X [MY, (Z P Z')] X'
- (X Y) [(Z P Z'), MQ] (X Y)'
- (X Y) [MZ, (P Q P')] (X Y)'
- (X2 Y) [(Z P Z'), MQ] (X2 Y)'
- X2 (MY Z) [(P Q P'), MY] (MY Z)' X2
- MX (Y2 Z) [MP, (Q V Q')] (Y2 Z)' MX'
- (X Y2 MZ P2 X) MY2 (X' Y2 MZ P2 X') MY2
- [X2, MY]
- [X Y Z X' Y', MP MQ]
- [MX, Y2]
- [X2, MY2]
- [MX2, Y2]
- [X Y Z X' Y', MZ]
- [MX, Y Z X Y' Z']
- [X Y Z X' Y', MZ2]
- [MX2, Y Z X Y' Z']
- [(X Y X') (Z P Z'), Q]
- [X, (Y Z Y') (P Q P')]
- [(X Y X') (Z P Z'), Q2]
- [X2, (Y Z Y') (P Q P')]
- [(X Y X'), Z]
- [X, (Y Z Y')]
- [X MY X', MZ]
- [MX Y MX', MZ]
- [(X Y X'), Z2]
- [X2, (Y Z Y')]
- X2 [(Y Z Y'), P2] X2
- X2 [Y2, (Z P Z')] X2
- X MY X'
- X MY2 X'
- MX Y MX'
- MX Y2 MX'
- X MY X MY'
- X MY X2 MY'
- (X Y) MZ (X Y)'
- (X Y) MZ2 (X Y)'
- X (Y Z) MP (Y Z)'
- X2 (Y Z) MP (Y Z)'
- (X MY X') (Z MP Z')
- (X MY2 X') (Z MP Z')
- [X, Y] [Z2, P]
- [X, Y2] [Z, P]
- X (MY MZ MP MY' MZ')
- X2 (MY MZ MP MY' MZ')
- (MX Y2 MX') MZ (MX Y2 MX') MZ'
- (MX Y2 MX') MZ2 (MX Y2 MX') MZ2
- X (MY Z2 MY') MP2 (MY Z2 MY') MP2
- X2 (MY Z2 MY') MP2 (MY Z2 MY') MP2
- (MX Y2) MZ2 (MX Y2)'
- X ((MY Z2) MP2 (MY Z2)')
- (X Y Z) [(P Q P'), MV] (X Y Z)'
- (X Y Z) [MP, (Q V Q')] (X Y Z)'
- [MX, Y MZ Y']
- [MX, MY Z MY']
- (X2 Y) [MZ, (P Q P')] (X2 Y)'
- (MX Y Z) [(P Q P'), MV] (MX Y Z)'
- (MX Y Z) [MP, (Q V Q')] (MX Y Z)'
- (X MY Z) [(P Q P'), MV] (X MY Z)'
- (X Y MZ) [(P Q P'), MV] (X Y MZ)'
- (MX Y) (Z MP (Z' Q Z) MP' Q Z) (MX Y)'
- MX Y Z P MZ' P' Z' P MZ P' Y' MX'
- MX Y Z MP Z' P' Z MP' Z' P Y' MX'
- MX Y X' Y MZ Y' X Y MZ' Y2 MX'
- MX Y2 Z MY Z' Y Z MY' Z' Y MX'
- X Y Z MY' Z' Y' Z MY Z' X'
- X Y Z MP Z' Y' Z MP' Z' X'
- X Y MZ Y' Z' Y MZ' Y' Z X'
- X Y X MZ X' Y' X MZ' X2
- X Y MZ Y' P Y MZ' Y' P' X'
- X Y MZ Y' X' Y MZ' Y'
- X Y MZ Y' X Y MZ' Y' X2
- X2 Y MZ Y' X Y MZ' Y' X
- X MY X' Z X MY' X' Z'
- X Y Z Y' MZ' Y Z' Y' MZ X'
- X MY Z Y' Z' MY' Z Y Z' X'
- (MX Y Z) [MZ', (Y' Z Y)] (MX Y Z)'
- (MX Y Z) [(Y' Z Y), MZ'] (MX Y Z)'
- (X Y Z) [(Z X Z'), MX'] (X Y Z)'
- (X Y Z) [MX', (Z X Z')] (X Y Z)'
- X Y2 Z MP Z' Y Z MP' Z' Y X'
- [X, MY MZ MY']
- [MX MY MX', Z]
- [X2, MY MZ MY']
- [MX MY MX', Z2]

Templates - Rotations
- CX CY
- CX CY CZ

Cube Layout
- ( ) Set To Default
- (●) Show Last Cube State - Numbers
- ( ) Show Last Cube State - Colors Only

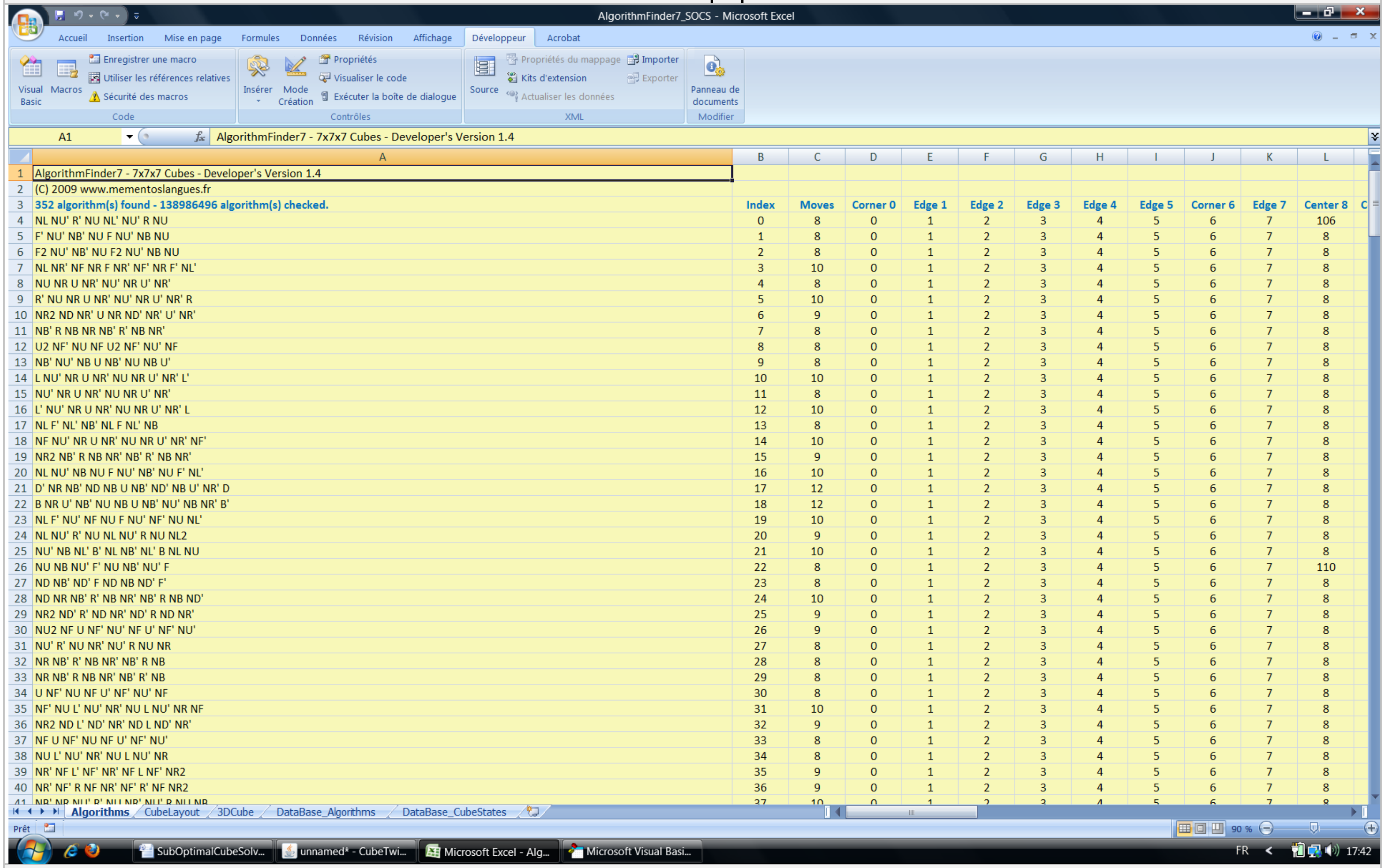
Mask Layout
- ( ) Set To Default
- (●) User Defined
- ( ) Set To -1 (excl. Face F)
- ( ) Set Last Layer To -1
- ( ) Set To -1 (excl. Face F Center)

Algorithms
- Show Shortest
- Run From Worksheet
- Generate SOCS DataBase

OK

- X MY MX' Z MX MY MX' Z' MX MY2 X'
- (MX Y Z) [P, MQ MV MQ'] (MX Y Z)'
- X MY2 MX' Z MX MY MX' Z' MX MY X'
- (MX Y Z) [MP MQ MP', V] (MX Y Z)'
- (MX MY MZ) [P, MQ MV MQ'] (MX MY MZ)'
- (MX MY MZ) [MP MQ MP', V] (MX MY MZ)'
- (X Y Z) [P, MQ MV MQ'] (X Y Z)'
- MX [Y, MZ MP MZ'] MX'
- MX [MY MZ MY', P] MX'
- (MX MY) [Z, MP MQ MP'] (MX MY)'
- (MX MY) [MZ MP MZ', Q] (MX MY)'
- MX MY Z MY' MP MY Z' MY' MP' MX'
- MX MY MZ P MZ' MY' MZ P' MZ' MX'
- (X MY) [MZ MP MZ', Q] (X MY)'
- (MX MY) [X, MZ MX MZ'] (MX MY)'
- (MX MY) [MZ MX MZ', X] (MX MY)'
- (MX MY) [Z, MX' MZ' MX] (MX MY)'
- (MX MY) [MX' MZ MX, Z'] (MX MY)'
- (X MY) [Z, MP MQ MP'] (X MY)'
- (X2 MY) [X, MZ MX' MZ'] (X2 MY)'
- MX [Y2, MZ MP MZ'] MX'
- MX [MY MZ MY', P2] MX'
- (MX MY) [Z2, MP MQ MP'] (MX MY)'
- (MX MY) [MZ MP MZ', Q2] (MX MY)'
- X [Y, MZ MP MZ'] X'
- X [MY MZ MY', P] X'
- (X2 MY) [MZ MX MZ', X'] (X2 MY)'
- X2 [Y, MZ MP MZ'] X2
- X2 [MY MZ MY', P] X2
- (X MY MZ) [P, MQ MV MQ'] (X MY MZ)'
- (X MY MZ) [MP MQ MP', V] (X MY MZ)'
- (X MY2) [Z, MP MQ MP'] (X MY2)'
- (X MY2) [MZ MP MZ', Q] (X MY2)'
- (X MY) [MX MZ MX', P] (X MY)'
- (X MY MZ MP) [Q, MV MA MV'] (X MY MZ MP)'
- (X MY) [Z, MX' MZ' MX] (X MY)'
- (X MY) [MX' MZ' MX, X] (X MY)'
- (X MY2) (MZ MP MQ V MQ' MP' MQ V' MQ' MZ') (X MY2)'
- (X MY) [Z, MX MP MX'] (X MY)'
- (X MY) [X', MZ MP MZ'] (X MY)'
- (X2 MY) [Z, MP MQ MP'] (X2 MY)'
- (X2 MY) [MZ MP MZ', Q] (X2 MY)'
- (X Y MZ) [P, MQ MV MQ'] (X Y MZ)'
- (X Y MZ) [MP MQ MP', V] (X Y MZ)'
- (X Y MZ) [P, MY' MX MY] (X Y MZ)'
- (X Y MZ) [MY' MX MY, P] (X Y MZ)'
- (X MY) [MZ MP MZ', X'] (X MY)'

7x7x7 Cube – A rather populated UserForm



AlgorithmFinder7 - 7x7x7 Cubes - Developer's Version 1.4

	A	B	C	D	E	F	G	H	I	J	K	L
1	AlgorithmFinder7 - 7x7x7 Cubes - Developer's Version 1.4											
2	(C) 2009 www.mementoslangues.fr											
3	**352 algorithm(s) found - 138986496 algorithm(s) checked.**	**Index**	**Moves**	**Corner 0**	**Edge 1**	**Edge 2**	**Edge 3**	**Edge 4**	**Edge 5**	**Corner 6**	**Edge 7**	**Center 8**
4	NL NU' R' NU NL' NU' R NU	0	8	0	1	2	3	4	5	6	7	106
5	F' NU' NB' NU F NU' NB NU	1	8	0	1	2	3	4	5	6	7	8
6	F2 NU' NB' NU F2 NU' NB NU	2	8	0	1	2	3	4	5	6	7	8
7	NL NR' NF NR F NR' NF' NR F' NL'	3	10	0	1	2	3	4	5	6	7	8
8	NU NR U NR' NU' NR U' NR'	4	8	0	1	2	3	4	5	6	7	8
9	R' NU NR U NR' NU' NR U' NR' R	5	10	0	1	2	3	4	5	6	7	8
10	NR2 ND NR' U NR ND' NR' U' NR'	6	9	0	1	2	3	4	5	6	7	8
11	NB' R NB NR NB' R' NB NR'	7	8	0	1	2	3	4	5	6	7	8
12	U2 NF' NU NF U2 NF' NU' NF	8	8	0	1	2	3	4	5	6	7	8
13	NB' NU' NB U NB' NU NB U'	9	8	0	1	2	3	4	5	6	7	8
14	L NU' NR U NR' NU NR U' NR' L'	10	10	0	1	2	3	4	5	6	7	8
15	NU' NR U NR' NU NR U' NR'	11	8	0	1	2	3	4	5	6	7	8
16	L' NU' NR U NR' NU NR U' NR' L	12	10	0	1	2	3	4	5	6	7	8
17	NL F' NL' NB' NL F NL' NB	13	8	0	1	2	3	4	5	6	7	8
18	NF NU' NR U NR' NU NR U' NR' NF'	14	10	0	1	2	3	4	5	6	7	8
19	NR2 NB' R NB NR' NB' R' NB NR'	15	9	0	1	2	3	4	5	6	7	8
20	NL NU' NB NU F NU' NB' NU F' NL'	16	10	0	1	2	3	4	5	6	7	8
21	D' NR NB' ND NB U NB' ND' NB U' NR' D	17	12	0	1	2	3	4	5	6	7	8
22	B NR U' NB' NU NB U NB' NU' NB NR' B'	18	12	0	1	2	3	4	5	6	7	8
23	NL F' NU' NF NU F NU' NF' NU NL'	19	10	0	1	2	3	4	5	6	7	8
24	NL NU' R' NU NL NU' R NU NL2	20	9	0	1	2	3	4	5	6	7	8
25	NU' NB NL' B' NL NB' NL' B NL NU	21	10	0	1	2	3	4	5	6	7	8
26	NU NB NU' F' NU NB' NU' F	22	8	0	1	2	3	4	5	6	7	110
27	ND NB' ND' F ND NB ND' F'	23	8	0	1	2	3	4	5	6	7	8
28	ND NR NB' R' NB NR' NB' R NB ND'	24	10	0	1	2	3	4	5	6	7	8
29	NR2 ND' R' ND NR' ND' R ND NR'	25	9	0	1	2	3	4	5	6	7	8
30	NU2 NF U NF' NU' NF U' NF' NU'	26	9	0	1	2	3	4	5	6	7	8
31	NU' R' NU NR' NU' R NU NR	27	8	0	1	2	3	4	5	6	7	8
32	NR NB' R' NB NR' NB' R NB	28	8	0	1	2	3	4	5	6	7	8
33	NR NB' R NB NR' NB' R' NB	29	8	0	1	2	3	4	5	6	7	8
34	U NF' NU NF U' NF' NU' NF	30	8	0	1	2	3	4	5	6	7	8
35	NF' NU L' NU' NR' NU L NU' NR NF	31	10	0	1	2	3	4	5	6	7	8
36	NR2 ND L' ND' NR' ND L ND' NR'	32	9	0	1	2	3	4	5	6	7	8
37	NF U NF' NU NF U' NF' NU'	33	8	0	1	2	3	4	5	6	7	8
38	NU L' NU' NR' NU L NU' NR	34	8	0	1	2	3	4	5	6	7	8
39	NR' NF L' NF' NR' NF L NF' NR2	35	9	0	1	2	3	4	5	6	7	8
40	NR' NF' R NF NR' NF' R' NF NR2	36	9	0	1	2	3	4	5	6	7	8

7x7x7 Cube – A selection of database algorithms – 3-cycle of centers – First orbit

Centers – Specific Templates

Centers – Orbits

There are 2 kinds of center orbits for an even-order cube and 4 kinds for an odd-order one:

1- Corner-Center, where each sticker is located at the intersection of two *equally* numbered slices, eg. NF/NR or N3F/N3R
2- Edge-Center or *obliques*, where each sticker is located at the intersection of two *differently* numbered slices, eg. NF/N3R
3- Midge-Center, where each sticker is located at the intersection of the middle edge slice and of any other slice, eg. NF/MR (odd-order cubes only)
4- True-Center, where each sticker is located on one of the 6 true center facelets (odd-order cubes only).

Centers – Templates

Generic (and slow to execute) templates are executed first, to give an idea of what short specific templates would look like. Then specific templates are selected on the following criteria:

1- No more than 3 variables per template, eg. F, R, U, to insure short execution time
2- No more than 12 moves per template, eg. 8, 9, 10, 11, 12 to insure short (and nearly optimal) algorithms

Centers – Corner-Center Orbits – Templates

A total of 21 pairs of fast specific templates is sufficient to find all algorithms of any corner-center orbit.

Centers – Corner-Center Orbits – Specific Templates – Niklas Commutators – 3-Cycles		
Algorithm Template	**Inverted Algorithm Template**	**Max. Number of Moves**
[X, MY MZ MY']	[MX MY MX', Z]	8
MX2 [MY2, MX Z MX'] MX2	MX2 [MX Y MX', MZ2] MX2	9
MX [Y, MZ MY' MZ] MX'	MX [MY MZ MY', Z'] MX'	10
MX [Y2, MZ MX MZ'] MX'	MX [MY MX MY', Z2] MX'	10
X [MX', MY Z MY'] X'	X [MY Z MY', MX'] X'	10
MX [X2, MY MZ MY'] MX'	MX [MY MZ MY', X2] MX'	10
MX [MY, MZ Y MZ'] MX'	MX [MY Z MY', MZ] MX'	10
MX [MX2, MY Z2 MY'] MX'	MX [MX2, MY Z2 MY'] MX'	10
X [MY, MZ Y2 MZ'] X'	X [MY Z2 MY', MZ] X'	10
(X MX') [MY, MX' Z2 MX] (X MX')'	(X MX') [MX' Y2 MX, MZ] (X MX')'	11
(X MY) [MZ, MY X2 MY'] (X MY)'	(X MY) [MY X2 MY', MZ] (X MY)'	11
(X MY) [Z2, MY MZ2 MY'] (X MY)'	(X MY) [MY MZ2 MY', Z2] (X MY)'	11
(X MY) [X2, MZ MX MZ'] (X MY)'	(X MY) [MZ MX MZ', X2] (X MY)'	12
(X MY) [Z2, MX MZ MX'] (X MY)'	(X MY) [MX MZ MX', Z2] (X MY)'	12
(X MY) [Z, MX MZ MX'] (X MY)'	(X MY) [MX MZ MX', Z] (X MY)'	12
(X MY) [X, MZ MX' MZ'] (X MY)'	(X MY) [MZ MX' MZ', X] (X MY)'	12
(X MY) [MZ, MY Z MY'] (X MY)'	(X MY) [MY Z MY', MZ] (X MY)'	12
(X MY) [Z, MY MZ MY'] (X MY)'	(X MY) [MY MZ MY', Z] (X MY)'	12
(MX Y) [MZ2, Y MX' Y'] (MX Y)'	(MX Y) [Y MX' Y', MZ2] (MX Y)'	12
(MX Y2) [MZ2, MX' Z2 MX] (MX Y2)'	(MX Y2) [MX' Z2 MX, MZ2] (MX Y2)'	12
(X2 Y) [MZ, MY' Z MY] (X2 Y)'	(X2 Y) [MY' Z MY, MZ] (X2 Y)'	12

Centers – Edge-Center Orbits – Templates

A total of 11 pairs of fast specific templates is sufficient to find all algorithms of any edge-center orbit.

Centers – Edge-Center Orbits – Specific Templates – Niklas Commutators – 3-Cycles		
Algorithm Template	**Inverted Algorithm Template**	**Max. Number of Moves**
[X, MY MZ MY']	[MX MY MX', Z]	8
MX2 [MY2, MX Z MX'] MX2	MX2 [MX Y MX', MZ2] MX2	9
MX [Y, MZ MY' MZ] MX'	MX [MY MZ MY', Z'] MX'	10
MX [Y2, MZ MX MZ'] MX'	MX [MY MX MY', Z2] MX'	10
X [MX', MY Z MY'] X'	X [MY Z MY', MX'] X'	10
(X MY) [X2, MZ MX MZ'] (X MY)'	(X MY) [MZ MX MZ', X2] (X MY)'	12
(X MY) [Z2, MX MZ MX'] (X MY)'	(X MY) [MX MZ MX', Z2] (X MY)'	12
(X MY) [Z, MX MZ MX'] (X MY)'	(X MY) [MX MZ MX', Z] (X MY)'	12
(X MY) [X, MZ MX' MZ'] (X MY)'	(X MY) [MZ MX' MZ', X] (X MY)'	12
(X MY) [MZ, MY Z MY'] (X MY)'	(X MY) [MY Z MY', MZ] (X MY)'	12
(X MY) [Z, MY MZ MY'] (X MY)'	(X MY) [MY MZ MY', Z] (X MY)'	12

Edges – Specific Templates

Edges – Templates

Half-turns have been added to the basic sets of quarter-turns to increase the number of available algorithms:

1- 1st set of 12 outer layer quarter-turns:
   X = {F2, R2, U2, L2, D2, B2, F, F', R, R', U, U', L, L', D, D', B, B'}
2- 2nd set of 12 outer layer quarter-turns:
   X' = { F2, R2, U2, L2, D2, B2, F', F, R', R, U', U, L', L, D', D, B', B}
3- Set of 6 outer layer half-turns:
   X2 = {F2, R2, U2, L2, D2, B2}
4- 1st set of 12 slice quarter-turns:
   MX = {NF2, NR2, NU2, NL2, ND2, NB2, NF, NF', NR, NR', NU, NU', NL, NL', ND, ND', NB, NB'}
5- 2nd set of 12 slice quarter-turns:
   MX' = {NF2, NR2, NU2, NL2, ND2, NB2, NF', NF, NR', NR, NU', NU, NL', NL, ND', ND, NB', NB}
6- Set of 6 slice half-turns:
   MX2 = {NF2, NR2, NU2, NL2, ND2, NB2}

Generic (and slow to execute) templates are executed first, to give an idea of what short specific templates would look like. Then specific templates are selected on the following criteria:

1- No more than 3 variables per template, eg. F, R, U, to insure short execution time
2- No more than 12 moves per template, eg. 8, 9, 10, 11, 12 to insure short (and nearly optimal) algorithms

A total of 15 pairs of fast specific templates is sufficient to find all algorithms of any edge orbit.

Edges – Specific Templates – Niklas Commutators – 3-Cycles		
Algorithm Template	**Inverted Algorithm Template**	**Max. Number of Moves**
[X Y X', MZ]	[MX, (Y Z Y')]	8
[X, Y MZ Y']	[X MY X', Z]	8
[X2, Y MZ Y']	[X MY X', Z2]	8
X [Y, X MZ X'] X'	X [X MY X', Z] X'	9
X [Y MZ Y', X] X'	X [X, Y MZ Y'] X'	9
MX [Y2, MX Z2 MX'] MX'	MX [MX Y2 MX', Z2] MX'	9
X [Y, Z MY' Z'] X'	X [Y MZ Y', Z'] X'	10
X [MY, Z Y' Z'] X'	X [Y Z Y', MZ'] X'	10
X [MY2, Z Y Z'] X'	X [Y Z Y', MZ2] X'	10
MX [Y2, Z MY2 Z'] MX'	MX [Y MZ2 Y', Z2] MX'	10
(MX Y) [Y, Z MY' Z'] (MX Y)'	(MX Y) [Z MY' Z', Y] (MX Y)'	11
(MX Y) [Z2, Y MZ2 Y'] (MX Y)'	(MX Y) [Y MZ2 Y', Z2] (MX Y)'	11
(X Y) [X2, Z MX' Z'] (X Y)'	(X Y) [Z MX' Z', X2] (X Y)'	12
(X MY) [Z2, X' MZ2 X] (X MY)'	(X MY) [X' MZ2 X, Z2] (X MY)'	12
(X Y2) [Z, X' MZ X] (X Y2)'	(X Y2) [X' MZ X, Z] (X Y2)'	12

Edges – Specific Templates – Odd-Order Cubes – Middle Edge Flips

Edges – Templates

Half-turns have been added to the basic sets of quarter-turns to increase the number of available algorithms:

1- 1st set of 12 outer layer quarter-turns:
   X = {F2, R2, U2, L2, D2, B2, F, F', R, R', U, U', L, L', D, D', B, B'}
2- 2nd set of 12 outer layer quarter-turns:
   X' = { F2, R2, U2, L2, D2, B2, F', F, R', R, U', U, L', L, D', D, B', B}
3- Set of 6 outer layer half-turns:
   X2 = {F2, R2, U2, L2, D2, B2}
4- 1st set of 12 slice quarter-turns:
   MX = {NF2, NR2, NU2, NL2, ND2, NB2, NF, NF', NR, NR', NU, NU', NL, NL', ND, ND', NB, NB'}
5- 2nd set of 12 slice quarter-turns:
   MX' = {NF2, NR2, NU2, NL2, ND2, NB2, NF', NF, NR', NR, NU', NU, NL', NL, ND', ND, NB', NB}
6- Set of 6 slice half-turns:
   MX2 = {NF2, NR2, NU2, NL2, ND2, NB2}

Generic (and slow to execute) templates are executed first, to give an idea of what short specific templates would look like. Then specific templates are selected on the following criteria:

1- No more than 3 variables per template, eg. F, R, U, to insure short execution time
2- No more than 16 moves per template, eg. 12 – 16 to insure short (and nearly optimal) algorithms

A total of 5 fast specific templates is sufficient to find all algorithms of any middle edge orbit.

Edges – Specific Templates – Middle-Edge Flips – 2-Cycles – MonoFlips	
Algorithm Template	**Max. Number of Moves**
[X Y Z X' Y', MZ]	12
[X Y Z X' Y', MZ']	12
(X) [Y Z P Y' Z', MP] (X)'	14
(X) [Y Z P Y' Z', MP'] (X)'	14
(X Y) [Z P Q Z' P', MQ] (X Y)'	16

Edges – Specific Templates – Parity Algorithms

Edges – Templates

Basic sets of quarter- and half-turns:

1- 1st set of 12 outer layer quarter-turns:
   X = {F, F', R, R', U, U', L, L', D, D', B, B'}
2- 2nd set of 12 outer layer quarter-turns:
   X' = { F', F, R', R, U', U, L', L, D', D, B', B}
3- Set of 6 outer layer half-turns:
   X2 = {F2, R2, U2, L2, D2, B2}
4- 1st set of 12 slice quarter-turns:
   MX = {NF, NF', NR, NR', NU, NU', NL, NL', ND, ND', NB, NB'}
5- 2nd set of 12 slice quarter-turns:
   MX' = {NF', NF, NR', NR, NU', NU, NL', NL, ND', ND, NB', NB}
6- Set of 6 slice half-turns:
   MX2 = {NF2, NR2, NU2, NL2, ND2, NB2}

Generic (and slow to execute) templates are executed first, to give an idea of what short specific templates would look like. Then specific templates are selected on the following criteria:

1- No more than 4 variables per template, eg. F, R, U, L to insure short execution time
2- No more than 18 moves per template, eg. 12 – 18 moves to insure short (and nearly optimal) algorithms

A total of 10 fast specific templates is sufficient to find all edge parity algorithms of any edge orbit.

Edges – Specific Templates – Parity Algorithms – 2-Cycles	
Double Edge Flip Algorithm Template	**Number of Moves**
(MX Y2 MX Y2) (Z2 MX Z2) (MP Y2 MP' Y2) MX2	12
(X) ((MY Z2 MY Z2) (P2 MY P2) (MQ Z2 MQ' Z2) MY2) (X')	14
(X2) ((MY Z2 MY Z2) (X2 MY X2) (MP Z2 MP' Z2) MY2) (X2)	14
(MX2 Y2) (MX Z2 MX' Z2) (P2 MX' P2) (MQ Z2 MQ' Z2) (MX2 Y2)'	15
(MX2 Y) (MZ P2 MZ' P2) (Q2 MZ' Q2) (MX P2 MX' P2) (MX2 Y)'	15
(X Y) ((MX Z2 MX Z2) (Y2 MX Y2) (MP Z2 MP' Z2) MX2) (X Y)'	16
(X2 Y) ((MX Z2 MX Z2) (Y2 MX Y2) (MP Z2 MP' Z2) MX2) (X2 Y)'	16
(X Y Z) ((MX P2 MX P2) (Y2 MX Y2) (MZ P2 MZ' P2) MX2) (X Y Z)'	18
(X Y Z) ((MP Y2 MP Y2) (Z2 MP Z2) (MX' Y2 MX Y2) MP2) (X Y Z)'	18
(X Y X) ((MZ P2 MZ P2) (Y2 MZ Y2) (MX P2 MX' P2) MZ2) (X Y X)'	18

Corners – Specific Templates

Corners – Templates

Half-turns have been added to the basic sets of quarter-turns to increase the number of available algorithms:

1- 1st set of 12 outer layer quarter-turns:
   X = {F2, R2, U2, L2, D2, B2, F, F', R, R', U, U', L, L', D, D', B, B'}
2- 2nd set of 12 outer layer quarter-turns:
   X' = { F2, R2, U2, L2, D2, B2, F', F, R', R, U', U, L', L, D', D, B', B}
3- Set of 6 outer layer half-turns:
   X2 = {F2, R2, U2, L2, D2, B2}

Generic (and slow to execute) templates are executed first, to give an idea of what short specific templates would look like. Then specific templates are selected on the following criteria:

1- No more than 3 variables per template, eg. F, R, U, to insure short execution time
2- No more than 12 moves per template, eg. 8, 9, 10, 11, 12 to insure short (and nearly optimal) algorithms

A total of 6 pairs of fast specific templates is sufficient to find all algorithms of the corner orbit.

Corners – Specific Templates – Niklas Commutators – 3-Cycles		
Algorithm Template	**Inverted Algorithm Template**	**Max. Number of Moves**
[X, (Y Z Y')]	[(X Y X'), Z]	8
X [Y, X Z2 X'] X'	X [X Y2 X', Z] X'	9
X [Y2, X Z X'] X'	X [X Y X', Z2] X'	9
(X2 Y) [X2, Y Z Y'] (X2 Y)'	(X2 Y) [Y Z Y', X2] (X2 Y)'	11
(X Y) [Y, X' Z X] (X Y)'	(X Y) [X' Z X, Y] (X Y)'	11
[X2, (Y Z2) Y' (Y Z2)']	[(X Y2) X' (X Y2)', Z2]	12

Corners – Specific Templates – Corner Twists

Corners – Templates

Half-turns have been added to the basic sets of quarter-turns to increase the number of available algorithms:

1- 1st set of 12 outer layer quarter-turns:
   X = {F2, R2, U2, L2, D2, B2, F, F', R, R', U, U', L, L', D, D', B, B'}
2- 2nd set of 12 outer layer quarter-turns:
   X' = { F2, R2, U2, L2, D2, B2, F', F, R', R, U', U, L', L, D', D, B', B}
3- Set of 6 outer layer half-turns:
   X2 = {F2, R2, U2, L2, D2, B2}

Generic (and slow to execute) templates are executed first, to give an idea of what short specific templates would look like. Then specific templates are selected on the following criteria:

1- No more than 4 variables per template, eg. F, R, U, L to insure short execution time
2- No more than 14 moves per template to insure short (and nearly optimal) algorithms

A total of 4 specific templates is sufficient to find all the corner twist algorithms.

Corners – Specific Templates – Corner Twists	
Algorithm Template	**Max. Number of Moves**
[X, (Y Z) P (Y Z)']	12
[(X Y) Z (X Y)', P]	12
X Y Z Y' P Z P' X' P Z' P' Y Z' Y'	14
X Y Z Y' X' Y P X' P' Z' P X P' Y'	14

Part III

Algorithm DataBase

DataBase Indexing

The simplest way of indexing algorithms for 3- and 2-cycles of corners, edges and centers is to define a database index as a function of integer numbers that will represent the cycle.

To look-up an algorithm that will cycle stickers, compute the index from simple formulas shown below and extract the indexed algorithm from database.

True Center Twist – Indexing

index_CenterTwist = B + 4xD + 16xL + 64xU + 256xR + 1024xF (Base 4 Numeral System)

where B, D, L, U, R and F are integer numbers ranging from 0 to 3 (0 = +0°, 1 = +90°, 2 = +180°, 3 = +270°), representing face center twists.

Index index_CenterTwist is an integer number ranging from 2 to 4095.

Because the sum of all true center twists must be equal to 0 or 2 Modulo 4, ie. 0° or 180° Modulo 360°, index_CenterTwist can't take more than 2047 values (trivial case not included).

True Center Twist – Indexing Example

B = 2 (+180°), D = 2 (+180°), L = 1 (+90°), U = 3 (+270°), R = 3 (+270°), F = 1 (+90°)

index_CenterTwist = 2 + 4x2 + 16x1 + 64x3 + 256x3 + 1024x1 = 2010

algorithm (2010) = T3B' T3F T3L T3R' T3F' T3U' T3D T3L T3R' T3B' T3F T3U T3L T3R' T3U' T3D T3L2 T3R2

Corner 3-Cycle – Indexing

index_Corner_3-Cycle = k + (j + ix24)x24 (Base 24 Numeral System)

where i, j and k are integer numbers ranging from 0 to 23, representing 3-cycles of corner stickers (i → j → k → i).

Index index_ Corner_3-Cycle is an integer number ranging from 78 to 13745.

Because the sum of 3 corner twists must be equal to 0 Modulo 3, index_Corner_3-Cycle can't take more than 9072 values.

Corner 3-Cycle – Indexing Example

(0 → 4 → 14 → 0)

index_Corner_3-Cycle = 14 + (4 + 0x24)x24 = 110

algorithm (110) = R U2 R D R' U2 R D' R2

Corner 2-Cycle – Indexing

Orbit Cube – Corner Numbering & Sticker Lettering								
Corner	0	1	2	3	4	5	6	7
Numbers	(0, 1, 2)	(3, 4, 5)	(6, 7, 8)	(9, 10, 11)	(12, 13, 14)	(15, 16, 17)	(18, 19, 20)	(21, 22, 23)
Letters	(D, O, F)	(A, E, I)	(B, L, S)	(C, R, P)	(W, J, H)	(X, G, N)	(U, M, Q)	(V, T, K)

index_Corner_2-Cycle = k + (j + ix8)x8 (Base 8 Numeral System) where i is an integer number ranging from 0 to 1:

1- If i = 0, then corner j is twisted clockwise and corner k counterclockwise
2- If i = 1, then corner j is twisted counterclockwise and corner k clockwise.

Integer numbers j and k range from 0 to 7 and represent each of the two corners.

Index index_Corner_2-Cycle is an integer number ranging from 1 to 126.

Because numbers i and j must be different, index_Corner_2-Cycle can’t take more than 112 values.

Corner 2-Cycle – Indexing Example

k = 3 (cw), j = 6 (ccw), i = 2

index_Corner_2-Cycle = 3 + (6 + 2x8)x8 = 106

algorithm (106) = U' L D' L' U L F U F' D F U' F' L'

Edge 3-Cycle – Indexing

index_Edge_3-Cycle = k + (j + ix24)x24 (Base 24 Numeral System)

where i, j and k are integer numbers ranging from 0 to 23, representing 3-cycles of edge stickers (i → j → k → i).

Index index_ Edge_3-Cycle is an integer number ranging from 26 to 13797.

Because stickers i, j and k must belong to 3 different edge pieces, index_Edge_3-Cycle can’t take all values between these two bounds.

Edge 3-Cycle – Indexing Example

(0 → 1 → 3 → 0)

index_Edge_3-Cycle = 3 + (1 + 0x24)x24 = 27

algorithm (27) = R' F U' NF' U F' U' NF U R

Edge 2-Cycle – Indexing

index_Edge_2-Cycle = j + ix24 (Base 24 Numeral System)

where j and i are integer numbers ranging from 0 to 23, representing middle slice edges. Midge pieces represented by stickers i and j are flipped.

Index index_Edge_2-Cycle is an integer number ranging from 1 to 574.

Because flipped edge pieces must be different, index_Edge_2-Cycle can’t take more than 528 values.

Edge 2-Cycle – Indexing Example

j = 8, i = 21

index_Edge_2-Cycle = 8 + 21x24 = 487

algorithm (487) = U' R F' U MR' U' F R' U F' MR F

Edge – Parity Fix – Indexing

index_Edge_ParityFix = j + ix24 (Base 24 Numeral System)

where j and i are integer numbers ranging from 0 to 23, representing edges. Two edge pieces represented by stickers i and j are permuted. Because it is a single 2-cycle of edge pieces, the permutation parity is odd.

Index index_Edge_ParityFix is an integer number ranging from 1 to 574.

Because permuted edge pieces must be different, index_Edge_ParityFix can’t take more than 528 values.

Edge – Parity Fix – Indexing Example

j = 4, i = 1

index_Edge_ParityFix = 4 + 1x24 = 28

algorithm (28) = NU2 R ND' B2 ND B2 R2 ND R2 NU' B2 NU B2 R' NU2

Corner-Center 3-Cycle – Indexing

index_Corner-Center_3-Cycle = k + (j + ix24)x24 (Base 24 Numeral System)

where i, j and k are integer numbers ranging from 0 to 23. These numbers represent 3-cycles of corner-center stickers (i → j → k → i).

Index index_Corner-Center_3-Cycle is an integer number ranging from 26 to 13797.

Because stickers i, j and k must belong to 3 different center pieces, index_Corner-Center_3-Cycle can’t take all values between these two bounds.

Corner-Center 3-Cycle – Indexing Example

(0 → 2 → 14 → 0)

index_Corner-Center_3-Cycle = 14 + (2 + 0x24)x24 = 62

algorithm (62) = NF' NU' NF' NU F2 NU' NF NU F2 NF

Edge-Center 3-Cycle – Indexing

index_Edge-Center_3-Cycle = k + (j + ix24)x24 (Base 24 Numeral System)

where i, j and k are integer numbers ranging from 0 to 23. These numbers represent 3-cycles of edge-center stickers (i → j → k → i).

Index index_Edge-Center_3-Cycle is an integer number ranging from 26 to 13797.

Because stickers i, j and k must belong to 3 different center pieces, index_Edge-Center_3-Cycle can’t take all values between these two bounds.

Edge-Center 3-Cycle – Indexing Example

(0 → 1 → 20 → 0)

index_Edge-Center_3-Cycle = 20 + (1 + 0x24)x24 = 44

algorithm (44) = N3R' F N3U NF' N3U' F' N3U NF N3U' N3R

DataBase Example

The computer program uses an internal *double* numbering system which has to be converted into a lettering system for interfacing to the Orbit Cube sticker notation. The conversion table between the two systems is shown below:

Conversion Table – Edge-Centers Numbering & Lettering – Orbit Cube																							
Cycle (i → j → k → i) or cycle (Sticker1 → Sticker2 → Sticker3 → Sticker1) in orbit '08'																							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
D	A	B	C	I	J	K	L	G	H	E	F	N	O	P	M	R	S	T	Q	W	X	U	V

The table below shows a partial list of algorithms for cycling edge-centers on a 7x7x7 cube. Stickers can be cycled either by 3 numbers (computer) or by 3 letters (Orbit Cube). Stickers are cycled in orbit '08' as follows:

cycle (i → j → k → i) or cycle (Sticker1 → Sticker2 → Sticker3 → Sticker1)

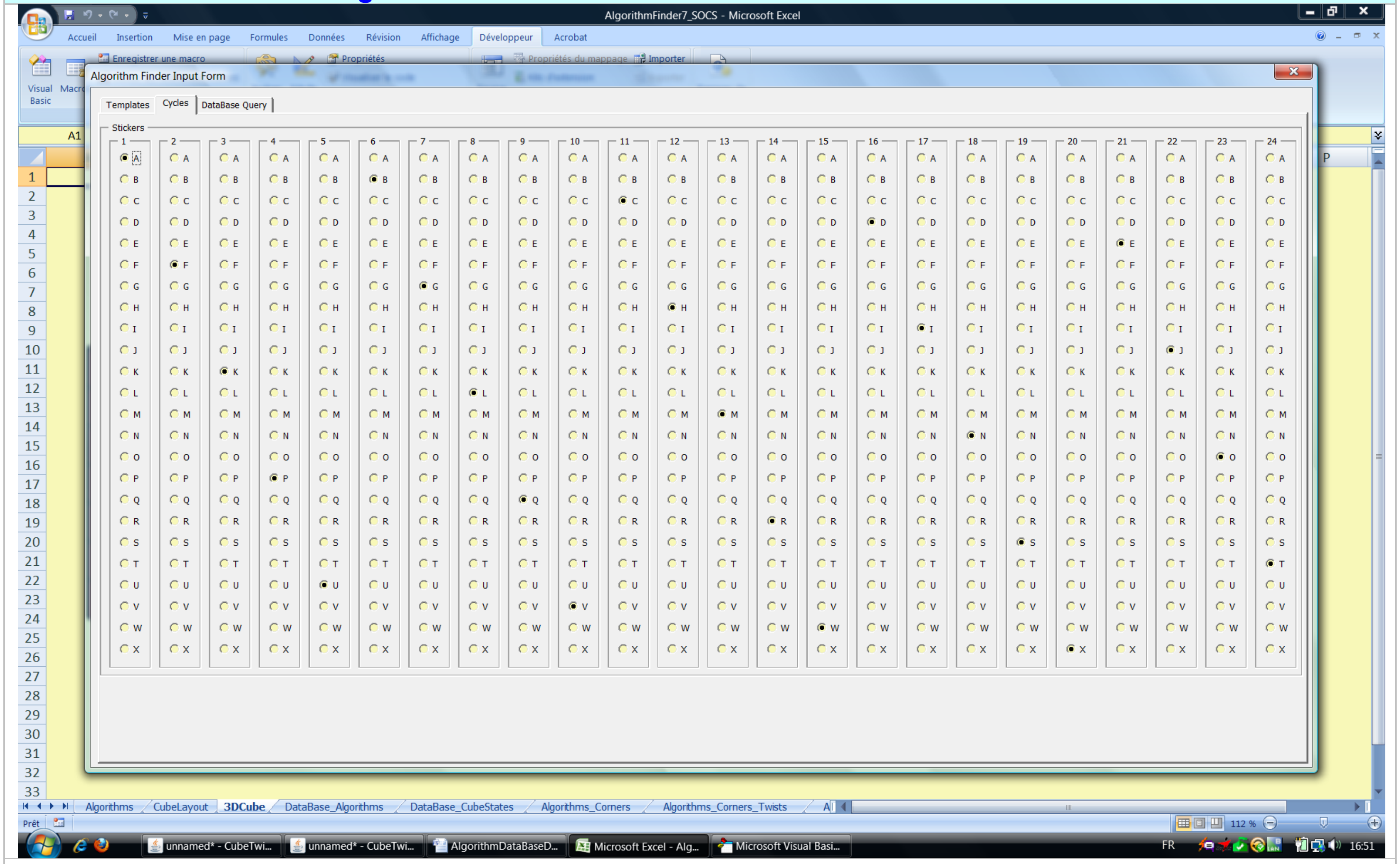
As shown in the table, algorithm number 33 will cycle stickers (D A H) and number 55 will cycle stickers (D B L).

DataBase Example – 3-cycles of Edge-Centers – Orbit '08'				
Algorithm	**Index**	**i/Sticker1**	**j/Sticker2**	**k/Sticker3**
N3L N3U NF N3U' F' N3U NF' N3U' F N3L'	26	0/D	1/A	2/B
N3D N3L NF N3L' F' N3L NF' N3L' F N3D'	27	0/D	1/A	3/C
F NR' N3F NR F' NR' N3F' NR	28	0/D	1/A	4/I
N3L' NB N3L F' N3L' NB' N3L F	29	0/D	1/A	5/J
F NR N3B NR' F' NR N3B' NR'	30	0/D	1/A	6/K
N3L NF N3L' F' N3L NF' N3L' F	31	0/D	1/A	7/L
F N3U' NB' N3U F' N3U' NB N3U	32	0/D	1/A	8/G
NU N3B NU' F' NU N3B' NU' F	33	0/D	1/A	9/H
F N3U NF' N3U' F' N3U NF N3U'	34	0/D	1/A	10/E
NU' N3F NU F' NU' N3F' NU F	35	0/D	1/A	11/F
F NR N3B' NR' F' NR N3B NR'	36	0/D	1/A	12/N
N3L NF' N3L' F' N3L NF N3L' F	37	0/D	1/A	13/O
F NR' N3F' NR F' NR' N3F NR	38	0/D	1/A	14/P
N3L' NB' N3L F' N3L' NB N3L F	39	0/D	1/A	15/M
F N3U NF N3U' F' N3U NF' N3U'	40	0/D	1/A	16/R
NU' N3F' NU F' NU' N3F NU F	41	0/D	1/A	17/S
F N3U' NB N3U F' N3U' NB' N3U	42	0/D	1/A	18/T
NU N3B' NU' F' NU N3B NU' F	43	0/D	1/A	19/Q
F N3R2 F' NU2 F N3R2 F' NU2	44	0/D	1/A	20/W
N3U2 F' NL2 F N3U2 F' NL2 F	45	0/D	1/A	21/X
F ND2 F' N3L2 F ND2 F' N3L2	46	0/D	1/A	22/U
NR2 F' N3D2 F NR2 F' N3D2 F	47	0/D	1/A	23/V
N3R N3U NF' N3U' F N3U NF N3U' F' N3R'	49	0/D	2/B	1/A
N3R N3D NF N3D' F' N3D NF' N3D' F N3R'	51	0/D	2/B	3/C
N3F' F2 ND' N3F' ND F2 ND' N3F ND N3F	52	0/D	2/B	4/I
NU' R' NU' N3R2 NU R NU' N3R2 NU2	53	0/D	2/B	5/J
N3B' NU N3B' NU' F2 NU N3B NU' F2 N3B	54	0/D	2/B	6/K
NU' R NU' N3R2 NU R' NU' N3R2 NU2	55	0/D	2/B	7/L
NB' F2 N3R' NB' N3R F2 N3R' NB N3R NB	56	0/D	2/B	8/G
N3L' U' N3L' ND2 N3L U N3L' ND2 N3L2	57	0/D	2/B	9/H
NF' N3L NF' N3L' F2 N3L NF N3L' F2 NF	58	0/D	2/B	10/E
N3L' U N3L' ND2 N3L U' N3L' ND2 N3L2	59	0/D	2/B	11/F
N3B' F2 ND N3B' ND' F2 ND N3B ND' N3B	60	0/D	2/B	12/N

Part IV

Algorithm Picker

Algorithm Picker 7 – Excel/VBA UserForm – 7x7x7 Cube


AlgorithmFinder7_SOCS - Microsoft Excel

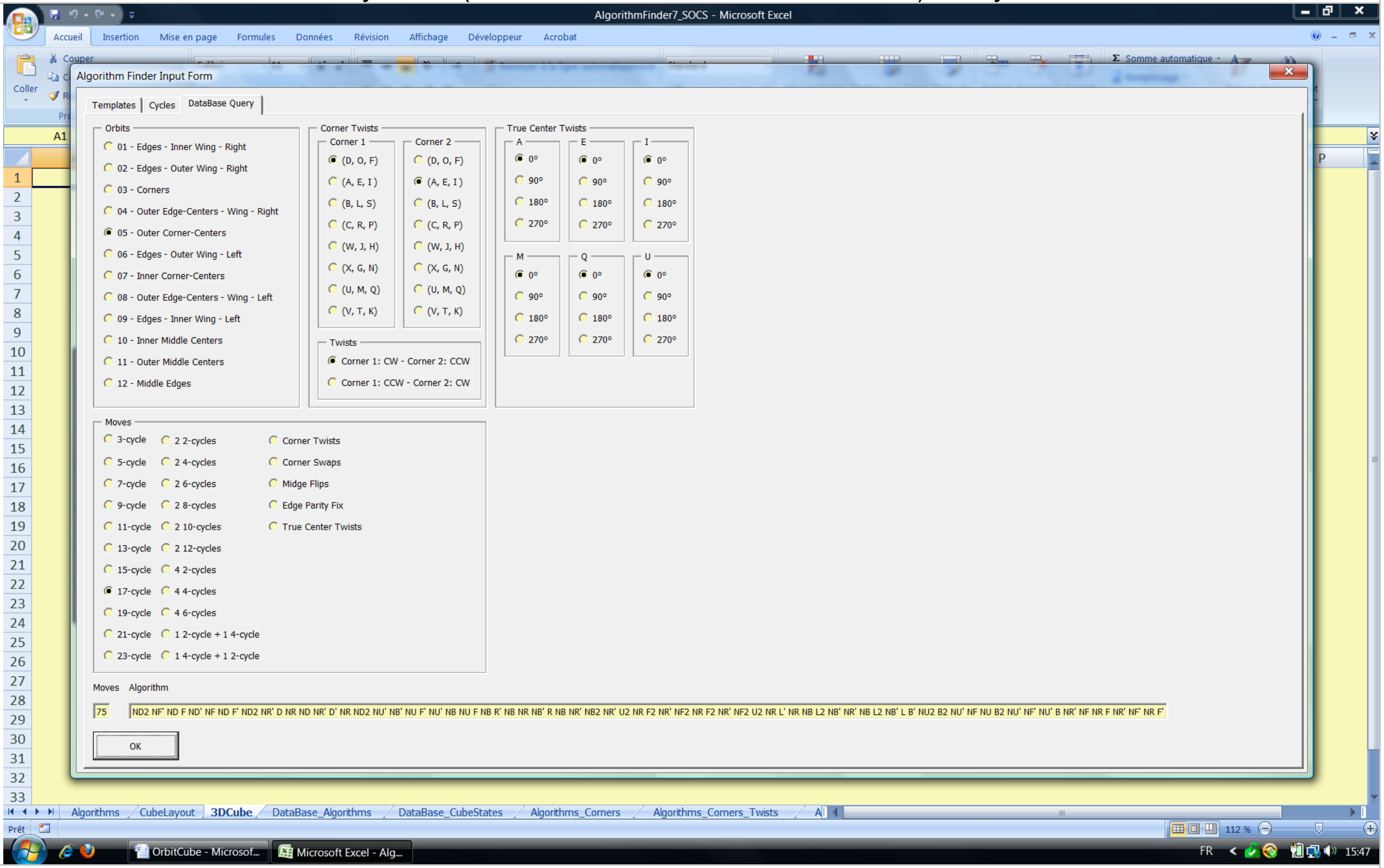
Algorithm Finder Input Form

Templates | Cycles | DataBase Query

Stickers

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
● A	F	K	P	U	B	G	L	Q	V	C	H	M	R	W	D	I	N	S	X	E	J	O	T

'Cycles' – (A F K P U B G L Q V C H M R W D I) 17-Cycle


AlgorithmFinder7_SOCS - Microsoft Excel

Algorithm Finder Input Form

Templates | Cycles | DataBase Query

Orbits: 01 - Edges - Inner Wing - Right; 02 - Edges - Outer Wing - Right; 03 - Corners; 04 - Outer Edge-Centers - Wing - Right; [✓ 05 - Outer Corner-Centers]; 06 - Edges - Outer Wing - Left; 07 - Inner Corner-Centers; 08 - Outer Edge-Centers - Wing - Left; 09 - Edges - Inner Wing - Left; 10 - Inner Middle Centers; 11 - Outer Middle Centers; 12 - Middle Edges

Corner Twists – Corner 1: [✓ (D, O, F)], (A, E, I), (B, L, S), (C, R, P), (W, J, H), (X, G, N), (U, M, Q), (V, T, K)

Corner Twists – Corner 2: (D, O, F), [✓ (A, E, I)], (B, L, S), (C, R, P), (W, J, H), (X, G, N), (U, M, Q), (V, T, K)

Twists: [✓ Corner 1: CW - Corner 2: CCW]; Corner 1: CCW - Corner 2: CW

True Center Twists: A [✓ 0°] 90° 180° 270°; E [✓ 0°] 90° 180° 270°; I [✓ 0°] 90° 180° 270°; M [✓ 0°] 90° 180° 270°; Q [✓ 0°] 90° 180° 270°; U [✓ 0°] 90° 180° 270°

Moves: 3-cycle; 5-cycle; 7-cycle; 9-cycle; 11-cycle; 13-cycle; 15-cycle; [✓ 17-cycle]; 19-cycle; 21-cycle; 23-cycle; 2 2-cycles; 2 4-cycles; 2 6-cycles; 2 8-cycles; 2 10-cycles; 2 12-cycles; 4 2-cycles; 4 4-cycles; 4 6-cycles; 1 2-cycle + 1 4-cycle; 1 4-cycle + 1 2-cycle; Corner Twists; Corner Swaps; Midge Flips; Edge Parity Fix; True Center Twists

Moves: 75

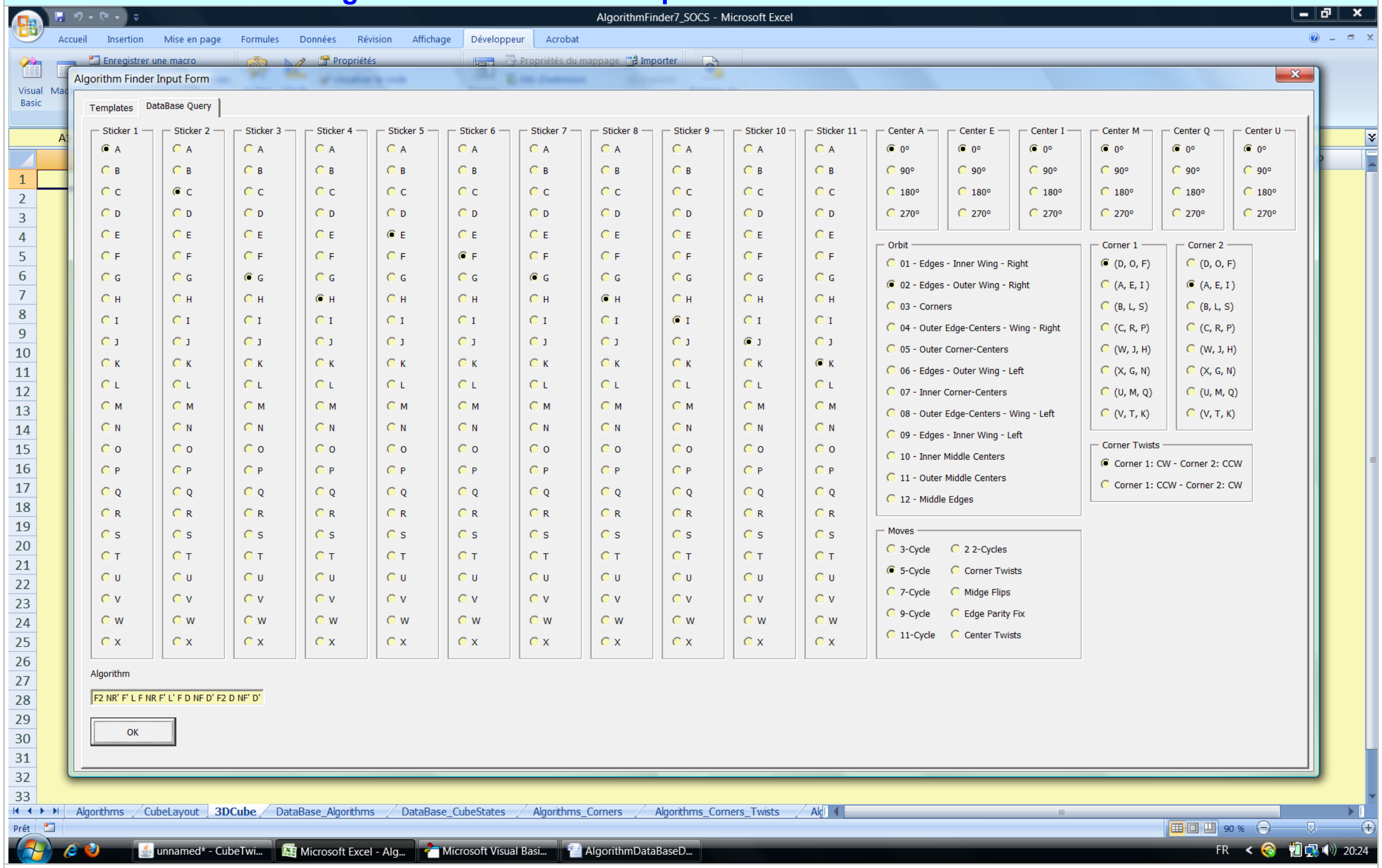
Algorithm: ND2 NF' ND F ND' NF ND F' ND2 NR' D NR ND NR' D' NR ND2 NU' NB' NU F' NU' NB NU F NB R' NB NR NB' R NB NR' NB2 NR' U2 NR F2 NR' NF2 NR F2 NR' NF2 U2 NR L' NR NB L2 NB' NR' NB L2 NB' L B' NU2 B2 NU' NF NU B2 NU' NF' NU' B NR' NF NR F NR' NF' NR F'

OK

'DataBase Query' – Orbit 05 – 17-Cycle (75 Moves)

Algorithm Picker 7 – Example 1

Algorithm Picker 7 – Example 1 – UserForm – 7x7x7 Cube



Algorithm Finder Input Form

Templates | DataBase Query

	Sticker 1	Sticker 2	Sticker 3	Sticker 4	Sticker 5	Sticker 6	Sticker 7	Sticker 8	Sticker 9	Sticker 10	Sticker 11
	[✓ A]	A	A	A	A	A	A	A	A	A	A
	B	B	B	B	B	B	B	B	B	B	B
	C	[✓ C]	C	C	C	C	C	C	C	C	C
	D	D	D	D	D	D	D	D	D	D	D
	E	E	E	E	[✓ E]	E	E	E	E	E	E
	F	F	F	F	F	[✓ F]	F	F	F	F	F
	G	G	[✓ G]	G	G	G	[✓ G]	G	G	G	G
	H	H	H	[✓ H]	H	H	H	[✓ H]	H	H	H
	I	I	I	I	I	I	I	I	[✓ I]	I	I
	J	J	J	J	J	J	J	J	J	[✓ J]	J
	K	K	K	K	K	K	K	K	K	K	[✓ K]
	L	L	L	L	L	L	L	L	L	L	L
	M	M	M	M	M	M	M	M	M	M	M
	N	N	N	N	N	N	N	N	N	N	N
	O	O	O	O	O	O	O	O	O	O	O
	P	P	P	P	P	P	P	P	P	P	P
	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
	R	R	R	R	R	R	R	R	R	R	R
	S	S	S	S	S	S	S	S	S	S	S
	T	T	T	T	T	T	T	T	T	T	T
	U	U	U	U	U	U	U	U	U	U	U
	V	V	V	V	V	V	V	V	V	V	V
	W	W	W	W	W	W	W	W	W	W	W
	X	X	X	X	X	X	X	X	X	X	X

Center A	Center E	Center I	Center M	Center Q	Center U
[✓ 0°]	[✓ 0°]	[✓ 0°]	[✓ 0°]	[✓ 0°]	[✓ 0°]
90°	90°	90°	90°	90°	90°
180°	180°	180°	180°	180°	180°
270°	270°	270°	270°	270°	270°

Orbit
- 01 - Edges - Inner Wing - Right
- [✓ 02 - Edges - Outer Wing - Right]
- 03 - Corners
- 04 - Outer Edge-Centers - Wing - Right
- 05 - Outer Corner-Centers
- 06 - Edges - Outer Wing - Left
- 07 - Inner Corner-Centers
- 08 - Outer Edge-Centers - Wing - Left
- 09 - Edges - Inner Wing - Left
- 10 - Inner Middle Centers
- 11 - Outer Middle Centers
- 12 - Middle Edges

Corner 1	Corner 2
[✓ (D, O, F)]	(D, O, F)
(A, E, I)	[✓ (A, E, I)]
(B, L, S)	(B, L, S)
(C, R, P)	(C, R, P)
(W, J, H)	(W, J, H)
(X, G, N)	(X, G, N)
(U, M, Q)	(U, M, Q)
(V, T, K)	(V, T, K)

Corner Twists
- [✓ Corner 1: CW - Corner 2: CCW]
- Corner 1: CCW - Corner 2: CW

Moves

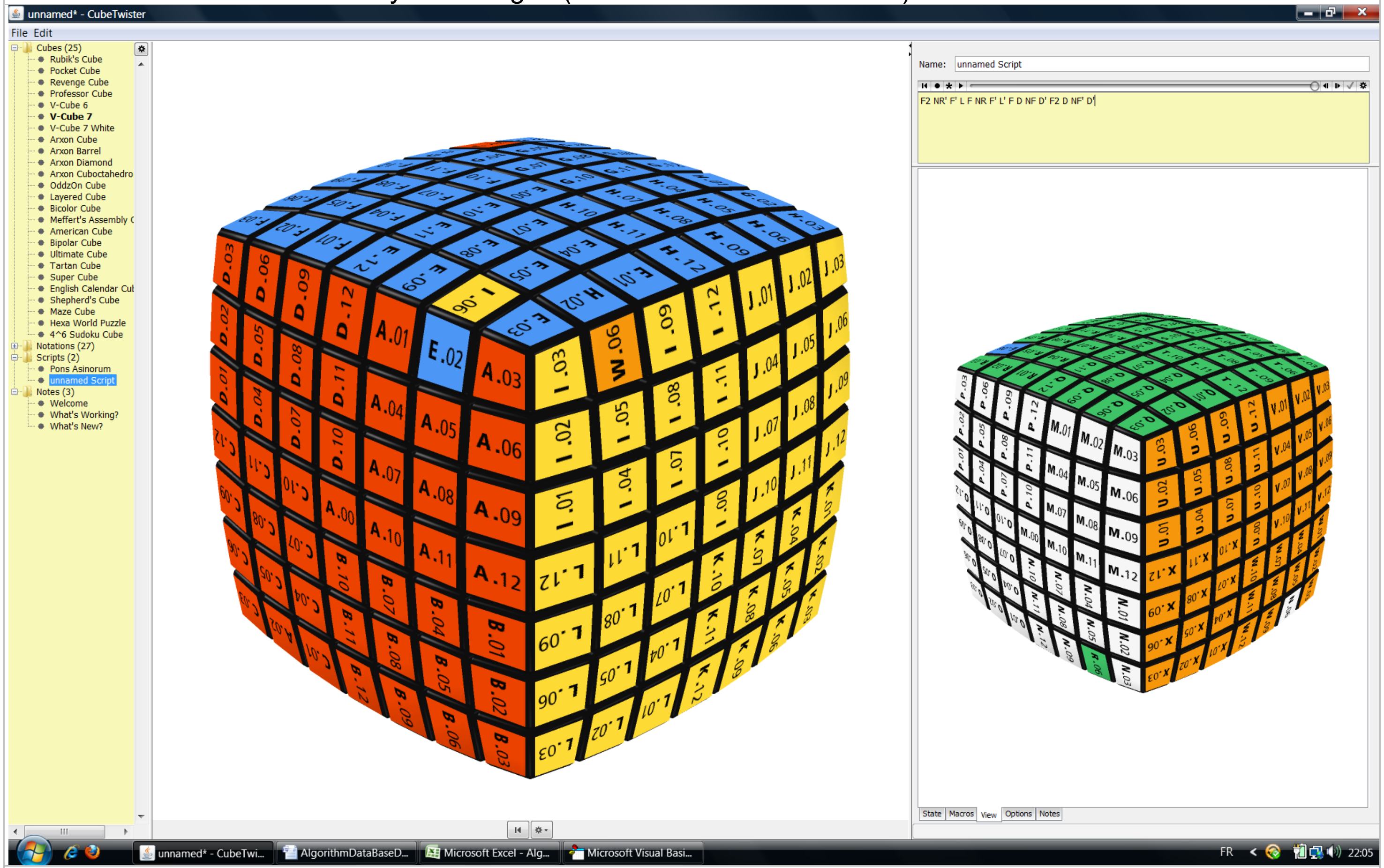
3-Cycle	2 2-Cycles
[✓ 5-Cycle]	Corner Twists
7-Cycle	Midge Flips
9-Cycle	Edge Parity Fix
11-Cycle	Center Twists

Algorithm

F2 NR' F' L F NR F' L' F D NF D' F2 D NF' D'

OK

5-Cycle of Edges (A → C → G → H → E → A) – Orbit 02



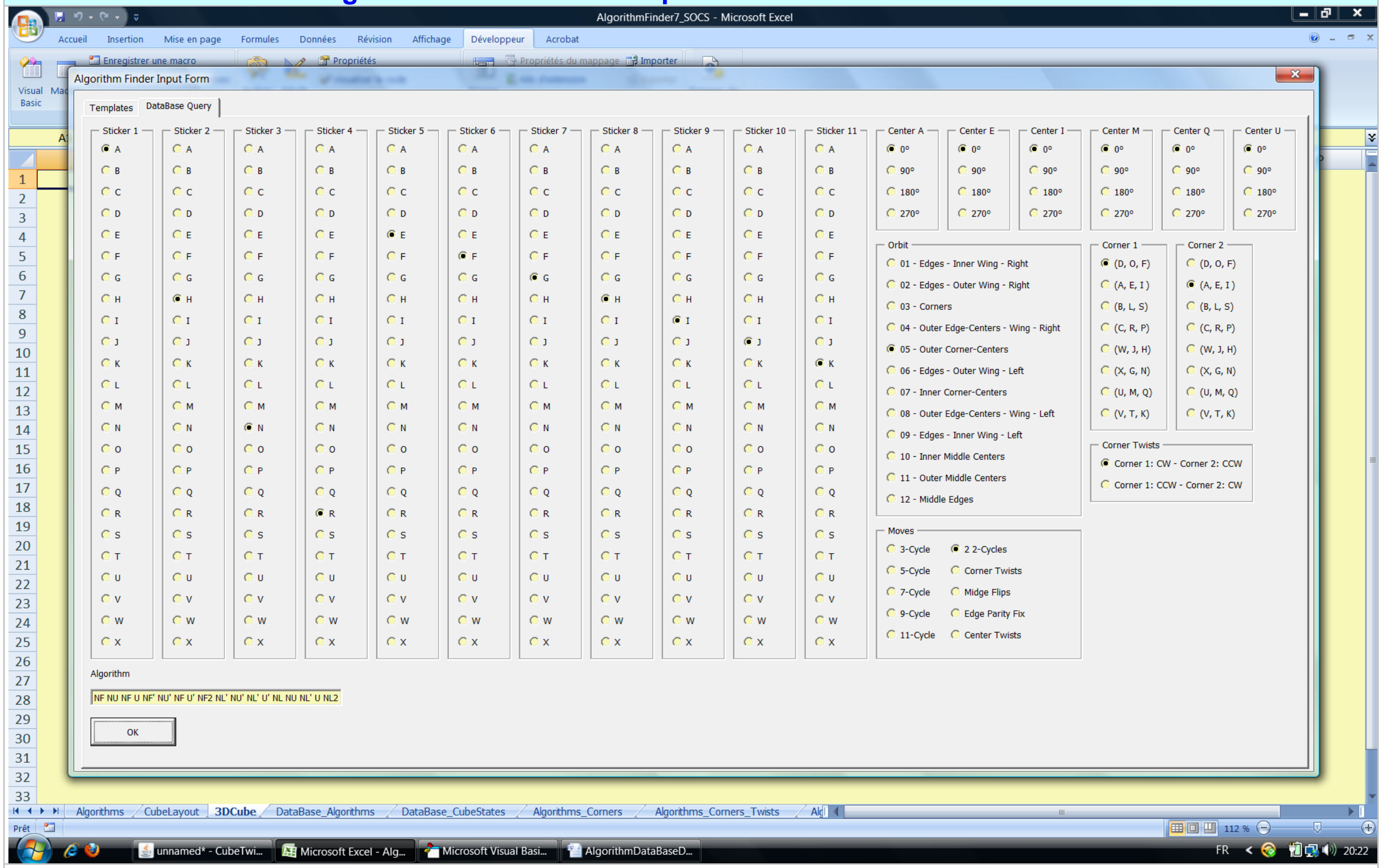
Name: unnamed Script

F2 NR' F' L F NR F' L' F D NF D' F2 D NF' D'

F2 NR' F' L F NR F' L' F D NF D' F2 D NF' D' (16 moves)

Algorithm Picker 7 – Example 2

Algorithm Picker 7 – Example 2 – UserForm – 7x7x7 Cube



Algorithm Finder Input Form

Templates | DataBase Query

Sticker 1	[✓ A]
Sticker 2	[✓ H]
Sticker 3	[✓ N]
Sticker 4	[✓ R]
Sticker 5	[✓ E]
Sticker 6	[✓ F]
Sticker 7	[✓ G]
Sticker 8	[✓ H]
Sticker 9	[✓ I]
Sticker 10	[✓ J]
Sticker 11	[✓ K]
Center A	[✓ 0°] 90° 180° 270°
Center E	[✓ 0°] 90° 180° 270°
Center I	[✓ 0°] 90° 180° 270°
Center M	[✓ 0°] 90° 180° 270°
Center Q	[✓ 0°] 90° 180° 270°
Center U	[✓ 0°] 90° 180° 270°

Orbit
- 01 - Edges - Inner Wing - Right
- 02 - Edges - Outer Wing - Right
- 03 - Corners
- 04 - Outer Edge-Centers - Wing - Right
- [✓ 05 - Outer Corner-Centers]
- 06 - Edges - Outer Wing - Left
- 07 - Inner Corner-Centers
- 08 - Outer Edge-Centers - Wing - Left
- 09 - Edges - Inner Wing - Left
- 10 - Inner Middle Centers
- 11 - Outer Middle Centers
- 12 - Middle Edges

Corner 1: [✓ (D, O, F)] (A, E, I) (B, L, S) (C, R, P) (W, J, H) (X, G, N) (U, M, Q) (V, T, K)

Corner 2: (D, O, F) [✓ (A, E, I)] (B, L, S) (C, R, P) (W, J, H) (X, G, N) (U, M, Q) (V, T, K)

Corner Twists: [✓ Corner 1: CW - Corner 2: CCW] Corner 1: CCW - Corner 2: CW

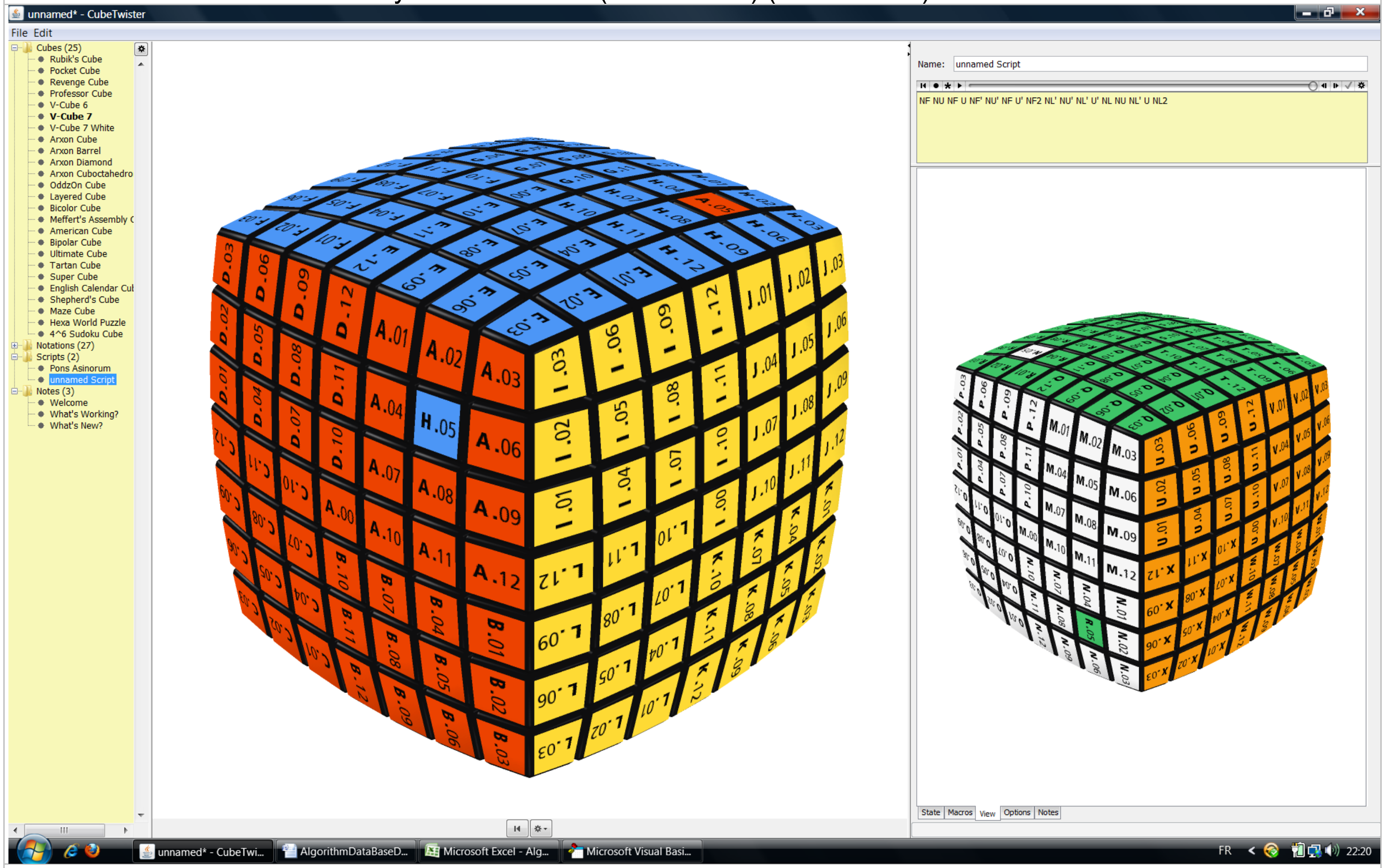
Moves: 3-Cycle [✓ 2 2-Cycles] 5-Cycle Corner Twists 7-Cycle Midge Flips 9-Cycle Edge Parity Fix 11-Cycle Center Twists

Algorithm

NF NU NF U NF' NU' NF U' NF2 NL' NU' NL' U' NL NU NL' U NL2

OK

2 2-Cycles of Centers (A → H → A) (N → R → N) – Orbit 05



unnamed* - CubeTwister

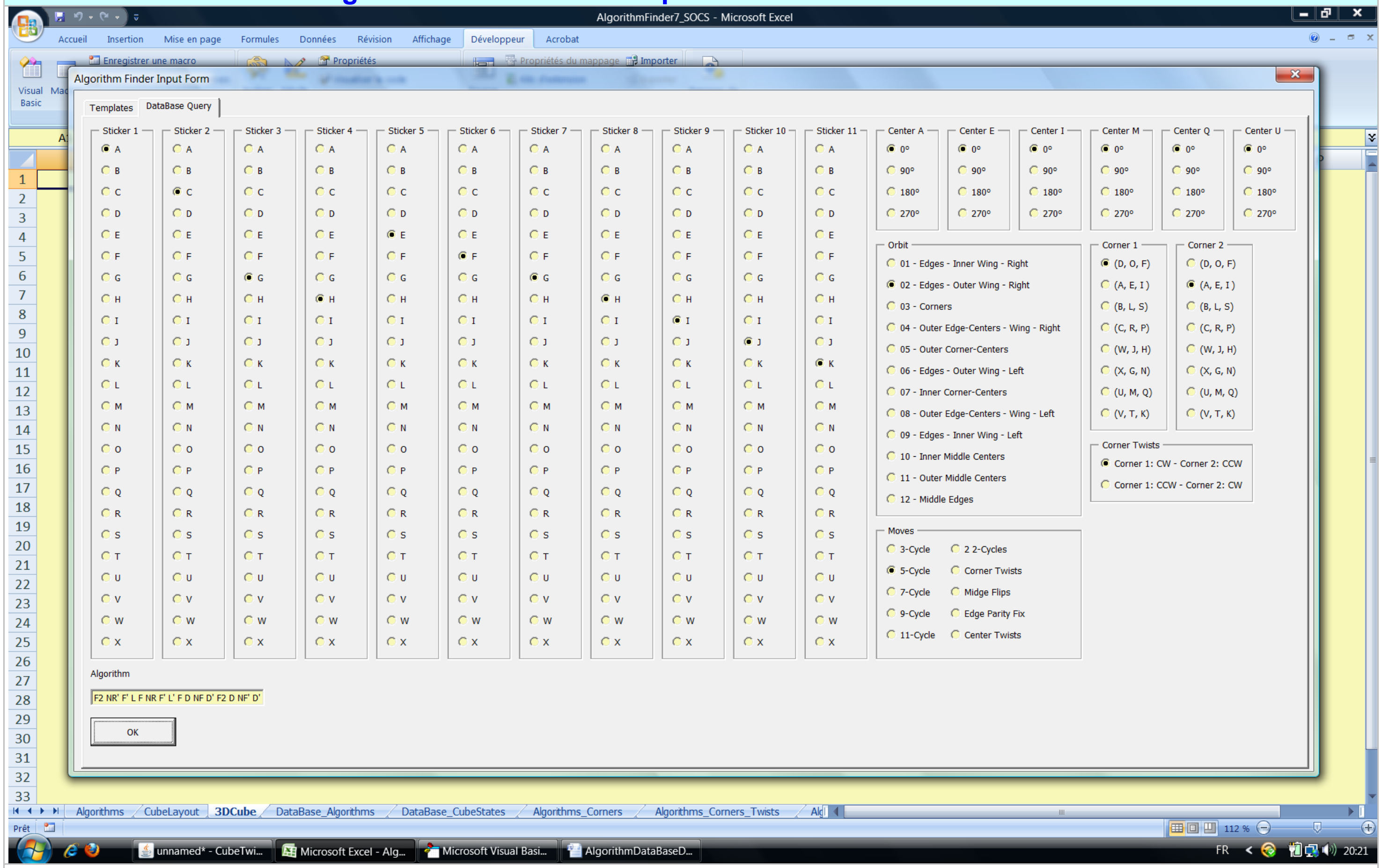
Name: unnamed Script

NF NU NF U NF' NU' NF U' NF2 NL' NU' NL' U' NL NU NL' U NL2

NF NU NF U NF' NU' NF U' NF2 NL' NU' NL' U' NL NU NL' U NL2 (18 moves)

Algorithm Picker 7 – Example 3

Algorithm Picker 7 – Example 3 – UserForm – 7x7x7 Cube



Algorithm Finder Input Form

Templates | DataBase Query

Sticker 1	Sticker 2	Sticker 3	Sticker 4	Sticker 5	Sticker 6	Sticker 7	Sticker 8	Sticker 9	Sticker 10	Sticker 11
A	C	G	H	E	F	G	H	I	J	K

Center A	Center E	Center I	Center M	Center Q	Center U
0°	0°	0°	0°	0°	0°

Orbit
- 01 - Edges - Inner Wing - Right
- [✓ 02 - Edges - Outer Wing - Right]
- 03 - Corners
- 04 - Outer Edge-Centers - Wing - Right
- 05 - Outer Corner-Centers
- 06 - Edges - Outer Wing - Left
- 07 - Inner Corner-Centers
- 08 - Outer Edge-Centers - Wing - Left
- 09 - Edges - Inner Wing - Left
- 10 - Inner Middle Centers
- 11 - Outer Middle Centers
- 12 - Middle Edges

Corner 1: [✓ (D, O, F)]

Corner 2: [✓ (A, E, I)]

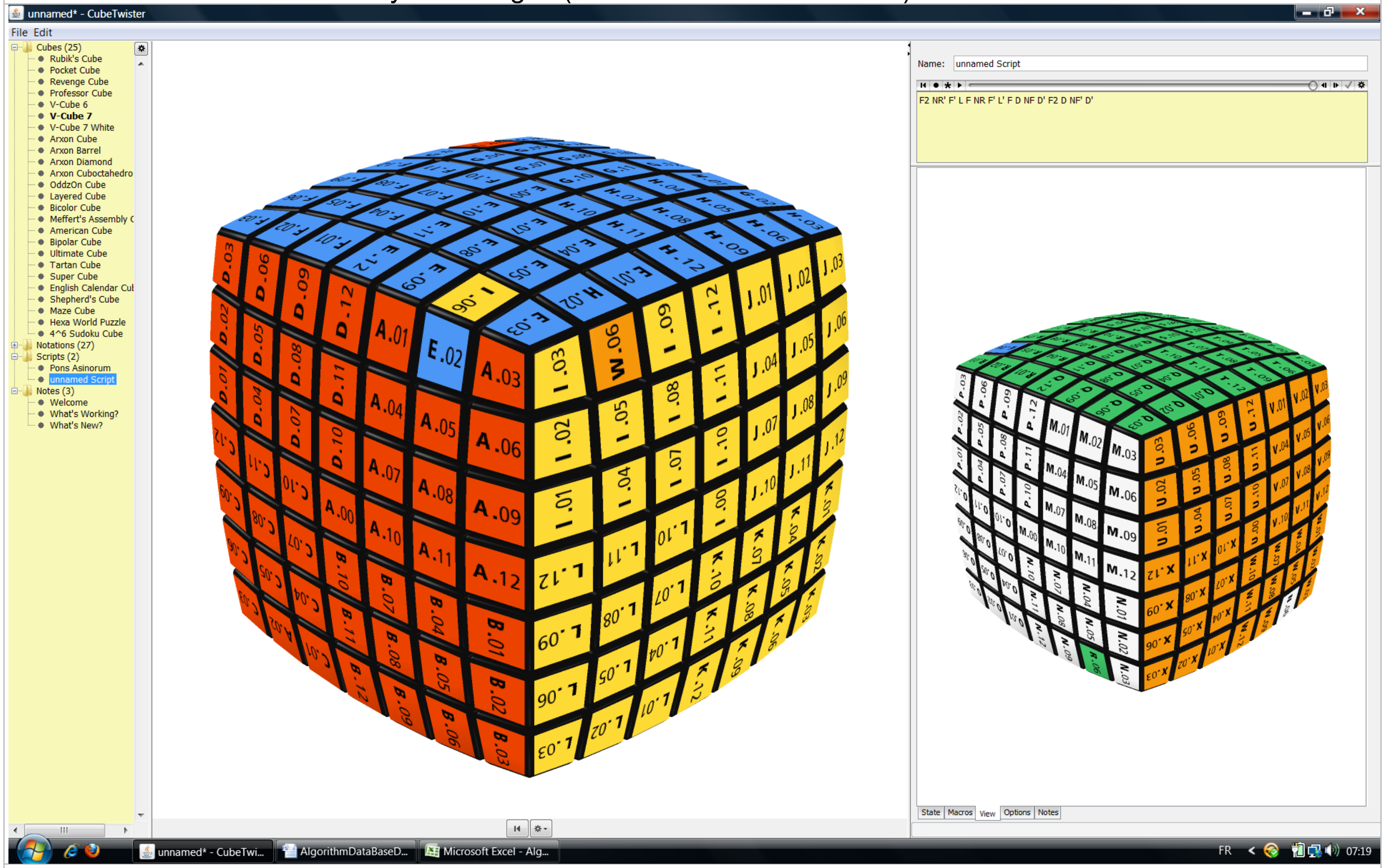
Corner Twists: [✓ Corner 1: CW - Corner 2: CCW] Corner 1: CCW - Corner 2: CW

Moves: 3-Cycle [✓ 5-Cycle] 7-Cycle 9-Cycle 11-Cycle 2 2-Cycles Corner Twists Midge Flips Edge Parity Fix Center Twists

Algorithm

F2 NR' F' L F NR F' L' F D NF D' F2 D NF' D'

5-Cycle of Edges (A → C → G → H → E → A) – Orbit 02



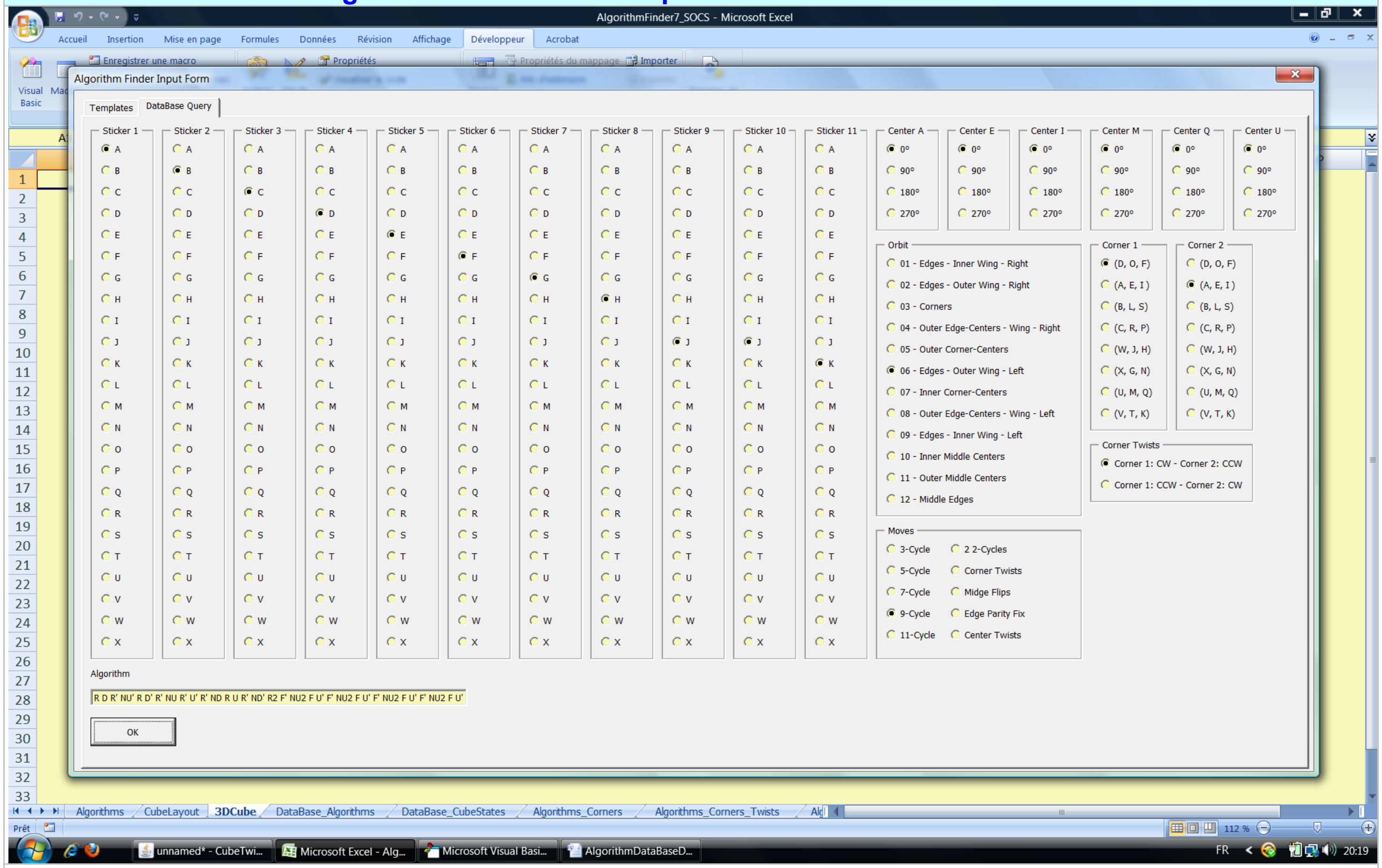
Name: unnamed Script

F2 NR' F' L F NR F' L' F D NF D' F2 D NF' D'

F2 NR' F' L F NR F' L' F D NF D' F2 D NF' D' (16 moves)

Algorithm Picker 7 – Example 4

Algorithm Picker 7 – Example 4 – UserForm – 7x7x7 Cube



AlgorithmFinder7_SOCS - Microsoft Excel

Algorithm Finder Input Form

Templates | DataBase Query

	Sticker 1	Sticker 2	Sticker 3	Sticker 4	Sticker 5	Sticker 6	Sticker 7	Sticker 8	Sticker 9	Sticker 10	Sticker 11
A	●										
B		●									
C			●								
D				●							
E					●						
F						●					
G							●				
H								●			
I											
J									●	●	
K											●
L											
M											
N											
O											
P											
Q											
R											
S											
T											
U											
V											
W											
X											

	Center A	Center E	Center I	Center M	Center Q	Center U
0°	●	●	●	●	●	●
90°						
180°						
270°						

Orbit
- 01 - Edges - Inner Wing - Right
- 02 - Edges - Outer Wing - Right
- 03 - Corners
- 04 - Outer Edge-Centers - Wing - Right
- 05 - Outer Corner-Centers
- [✓ 06 - Edges - Outer Wing - Left]
- 07 - Inner Corner-Centers
- 08 - Outer Edge-Centers - Wing - Left
- 09 - Edges - Inner Wing - Left
- 10 - Inner Middle Centers
- 11 - Outer Middle Centers
- 12 - Middle Edges

Corner 1	Corner 2
[✓ (D, O, F)]	(D, O, F)
(A, E, I)	[✓ (A, E, I)]
(B, L, S)	(B, L, S)
(C, R, P)	(C, R, P)
(W, J, H)	(W, J, H)
(X, G, N)	(X, G, N)
(U, M, Q)	(U, M, Q)
(V, T, K)	(V, T, K)

Corner Twists
- [✓ Corner 1: CW - Corner 2: CCW]
- Corner 1: CCW - Corner 2: CW

Moves

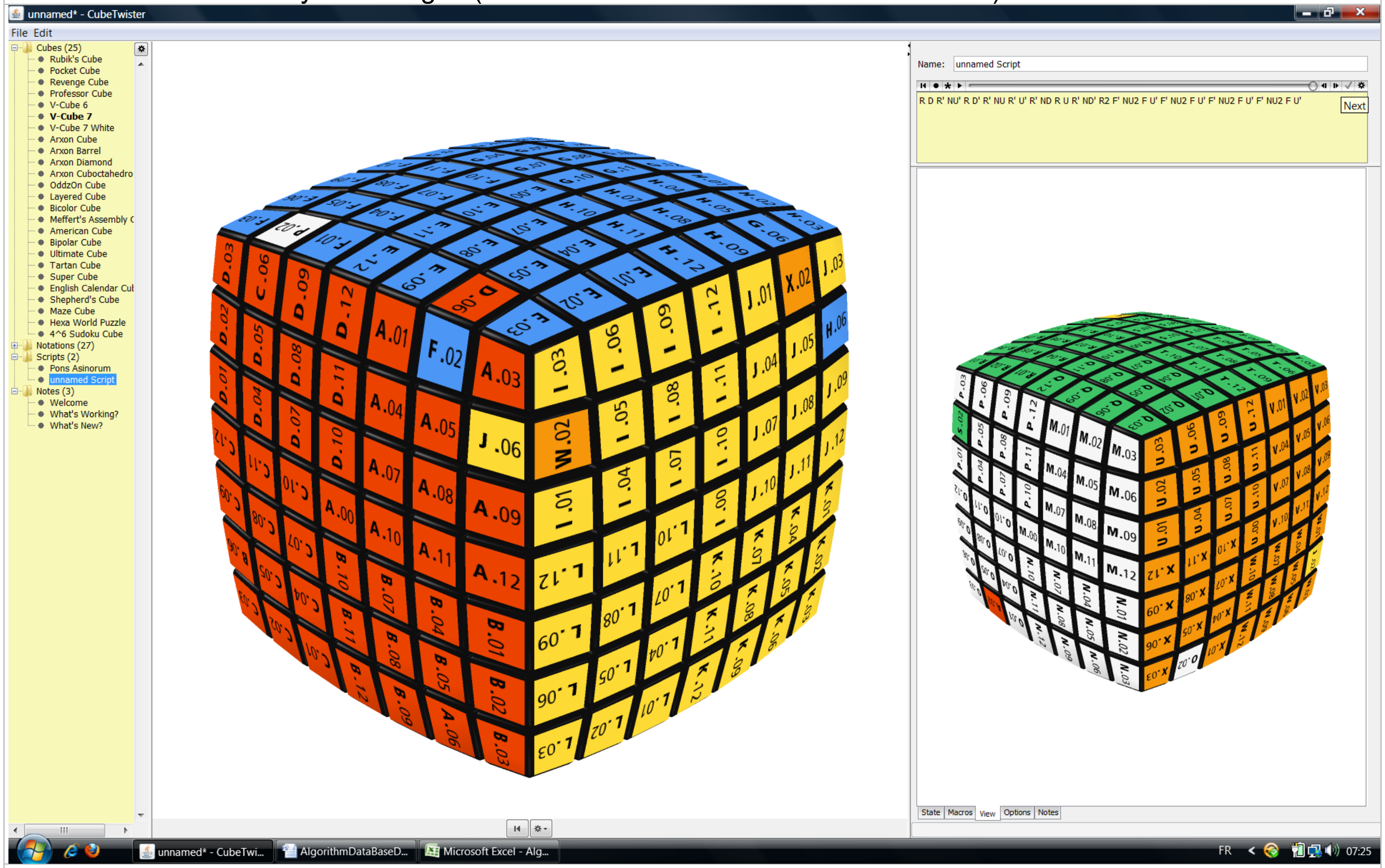
3-Cycle	2 2-Cycles
5-Cycle	Corner Twists
7-Cycle	Midge Flips
[✓ 9-Cycle]	Edge Parity Fix
11-Cycle	Center Twists

Algorithm

R D R' NU' R D' R' NU R' U' R' ND R U R' ND' R2 F' NU2 F U' F' NU2 F U' F' NU2 F U' F' NU2 F U'

OK

9-Cycle of Edges (A → B → C → D → E → F → G → H → J) – Orbit 06



unnamed* - CubeTwister

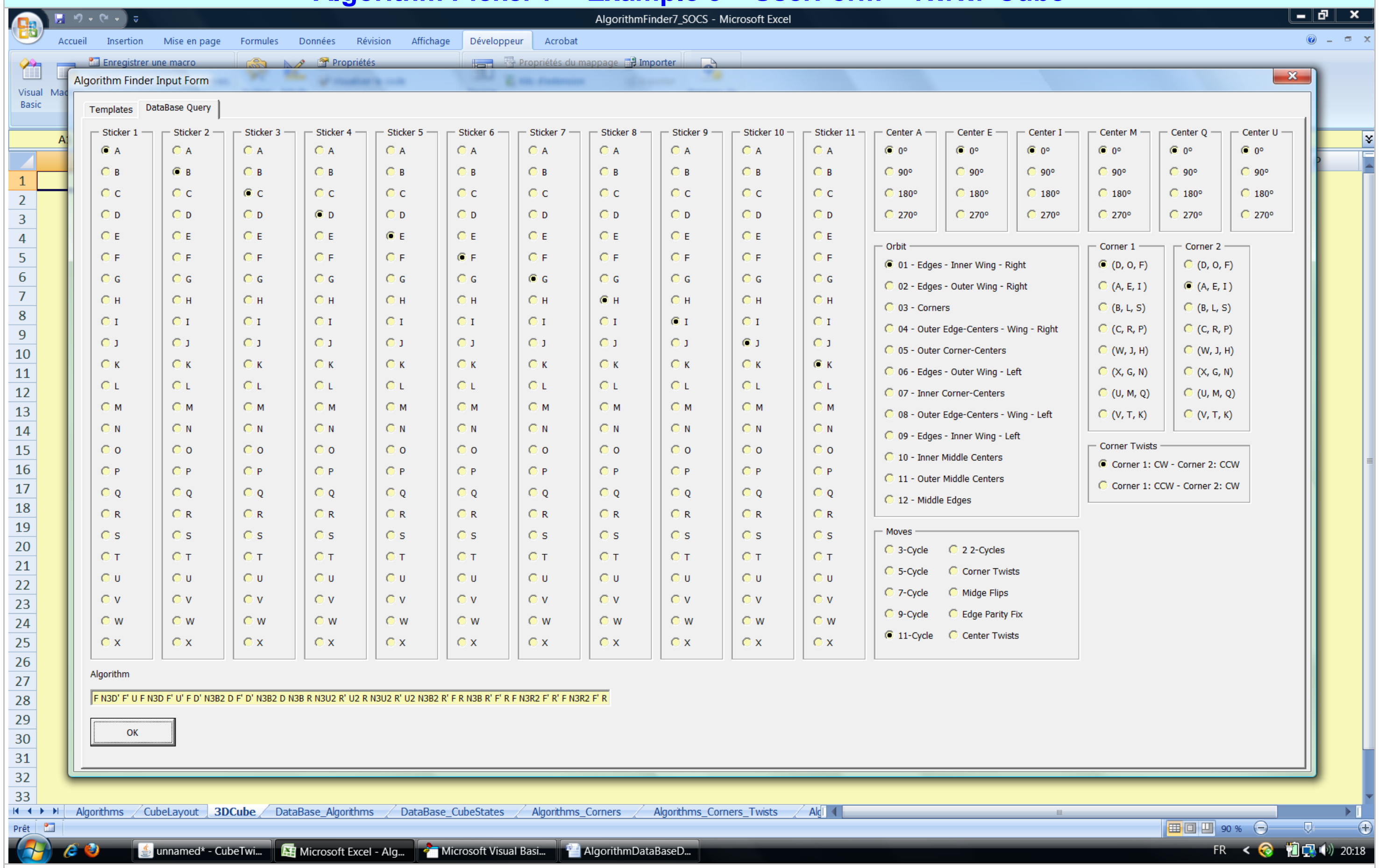
Name: unnamed Script

R D R' NU' R D' R' NU R' U' R' ND R U R' ND' R2 F' NU2 F U' F' NU2 F U' F' NU2 F U' F' NU2 F U'

R D R' NU' R D' R' NU R' U' R' ND R U R' ND' R2 F' NU2 F U' F' NU2 F U' F' NU2 F U' F' NU2 F U' (33 moves)

Algorithm Picker 7 – Example 5

Algorithm Picker 7 – Example 5 – UserForm – 7x7x7 Cube


AlgorithmFinder7_SOCS - Microsoft Excel

Algorithm Finder Input Form

Templates | DataBase Query

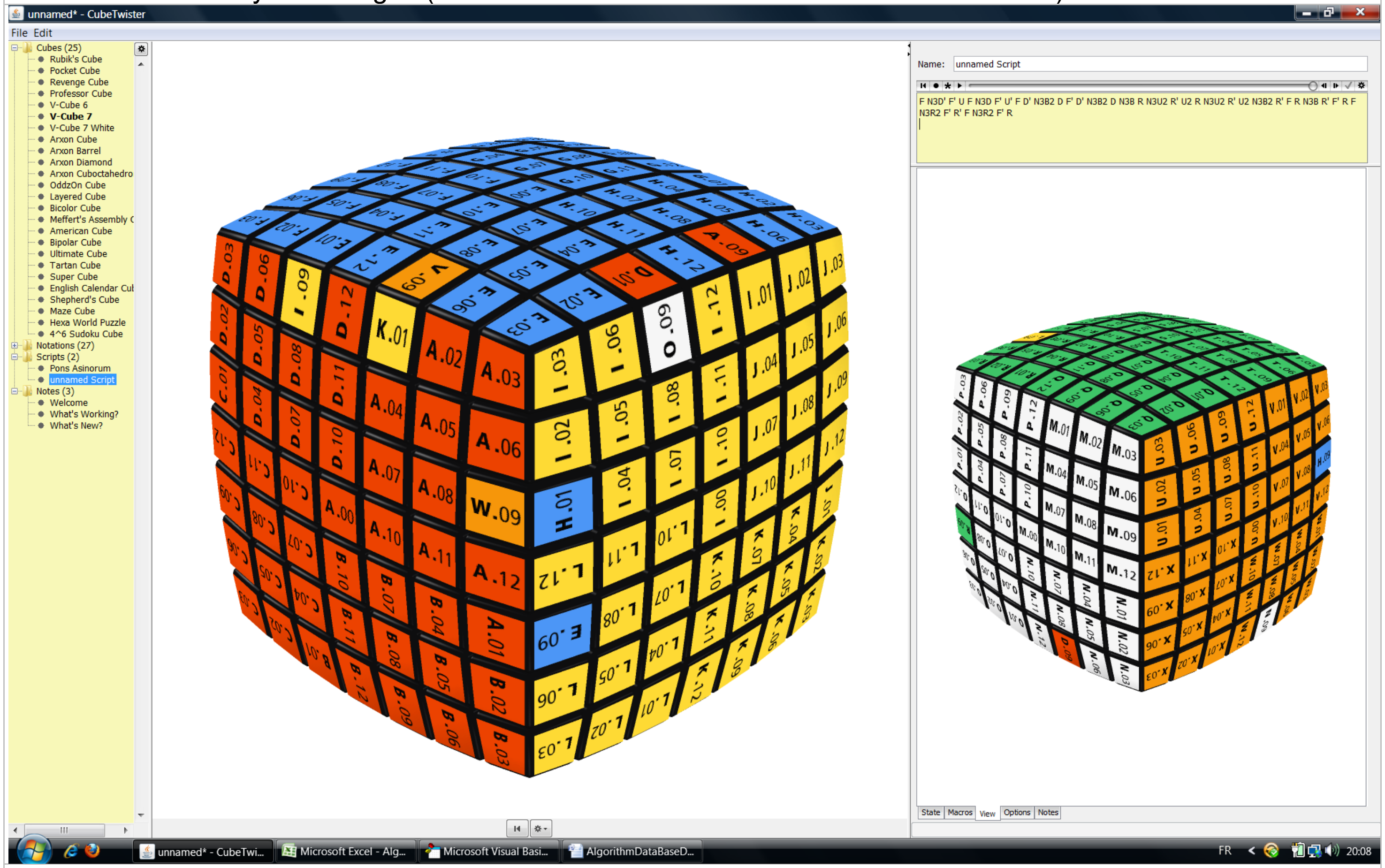
Sticker 1	[✓ A]
Sticker 2	[✓ B]
Sticker 3	[✓ C]
Sticker 4	[✓ D]
Sticker 5	[✓ E]
Sticker 6	[✓ F]
Sticker 7	[✓ G]
Sticker 8	[✓ H]
Sticker 9	[✓ I]
Sticker 10	[✓ J]
Sticker 11	[✓ K]
Center A	[✓ 0°]
Center E	[✓ 0°]
Center I	[✓ 0°]
Center M	[✓ 0°]
Center Q	[✓ 0°]
Center U	[✓ 0°]
Orbit	[✓ 01 - Edges - Inner Wing - Right]
Corner 1	[✓ (D, O, F)]
Corner 2	[✓ (A, E, I)]
Corner Twists	[✓ Corner 1: CW - Corner 2: CCW]
Moves	[✓ 11-Cycle]

Algorithm

F N3D' F' U F N3D F' U' F D' N3B2 D F' D' N3B2 D N3B R N3U2 R' U2 R N3U2 R' U2 N3B2 R' F R N3B R' F' R F N3R2 F' R' F N3R2 F' R

OK

11-Cycle of Edges (A → B → C → D → E → F → G → H → I → J → K) – Orbit 01


unnamed* - CubeTwister

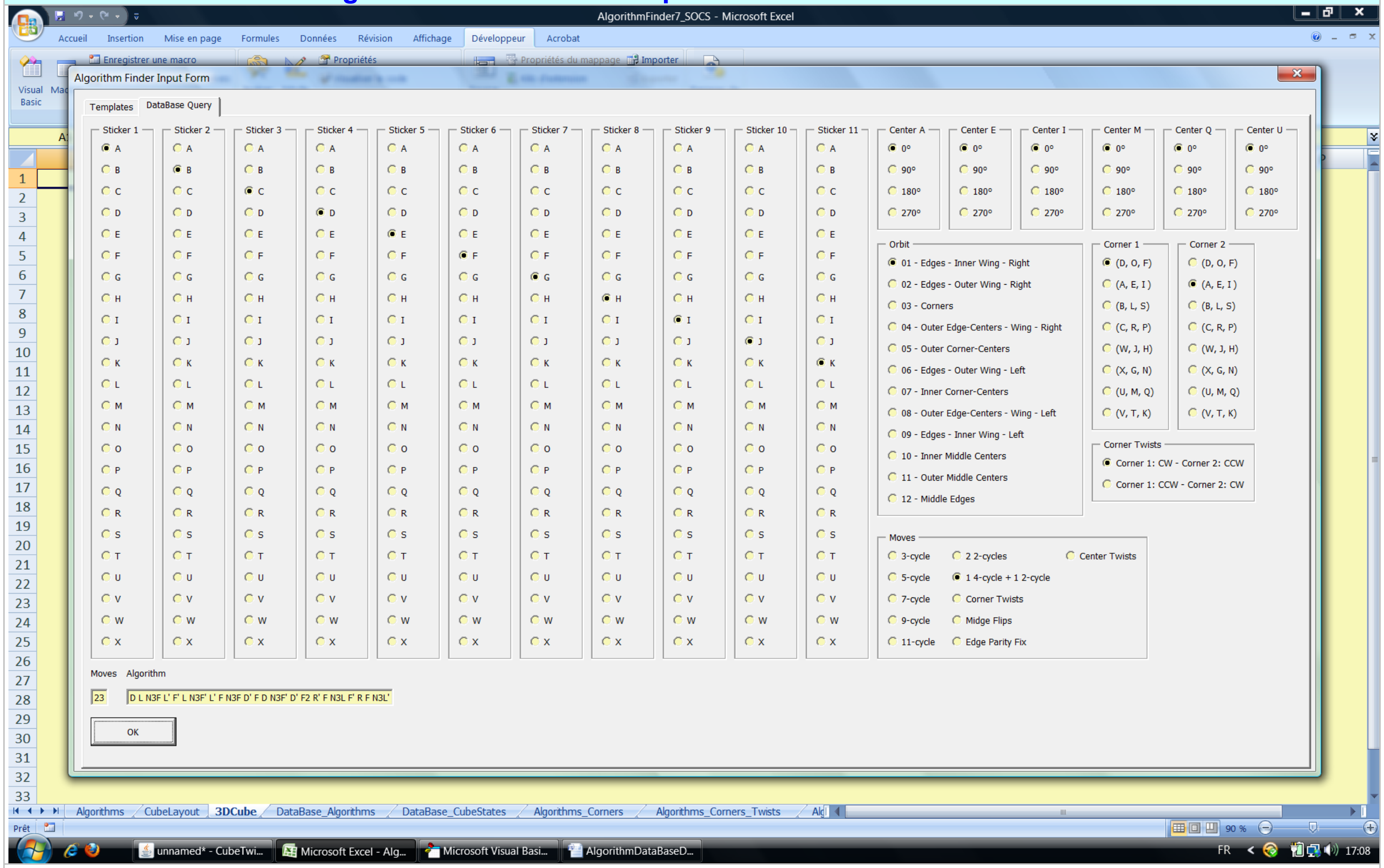
Name: unnamed Script

F N3D' F' U F N3D F' U' F D' N3B2 D F' D' N3B2 D N3B R N3U2 R' U2 R N3U2 R' U2 N3B2 R' F R N3B R' F' R F N3R2 F' R' F N3R2 F' R

F N3D' F' U F N3D F' U' F D' N3B2 D F' D' N3B2 D N3B R N3U2 R' U2 R N3U2 R' U2 N3B2 R' F R N3B R' F' R F N3R2 F' R' F N3R2 F' R (41 moves)

Algorithm Picker 7 – Example 6

Algorithm Picker 7 – Example 6 – UserForm – 7x7x7 Cube



Algorithm Finder Input Form

Templates | DataBase Query

Sticker 1	Sticker 2	Sticker 3	Sticker 4	Sticker 5	Sticker 6	Sticker 7	Sticker 8	Sticker 9	Sticker 10	Sticker 11
A	B	C	D	E	F	G	H	I	J	K

Center A	Center E	Center I	Center M	Center Q	Center U
0°	0°	0°	0°	0°	0°

Orbit
- [✓ 01 - Edges - Inner Wing - Right]
- 02 - Edges - Outer Wing - Right
- 03 - Corners
- 04 - Outer Edge-Centers - Wing - Right
- 05 - Outer Corner-Centers
- 06 - Edges - Outer Wing - Left
- 07 - Inner Corner-Centers
- 08 - Outer Edge-Centers - Wing - Left
- 09 - Edges - Inner Wing - Left
- 10 - Inner Middle Centers
- 11 - Outer Middle Centers
- 12 - Middle Edges

Corner 1: [✓ (D, O, F)] (A, E, I) (B, L, S) (C, R, P) (W, J, H) (X, G, N) (U, M, Q) (V, T, K)

Corner 2: (D, O, F) [✓ (A, E, I)] (B, L, S) (C, R, P) (W, J, H) (X, G, N) (U, M, Q) (V, T, K)

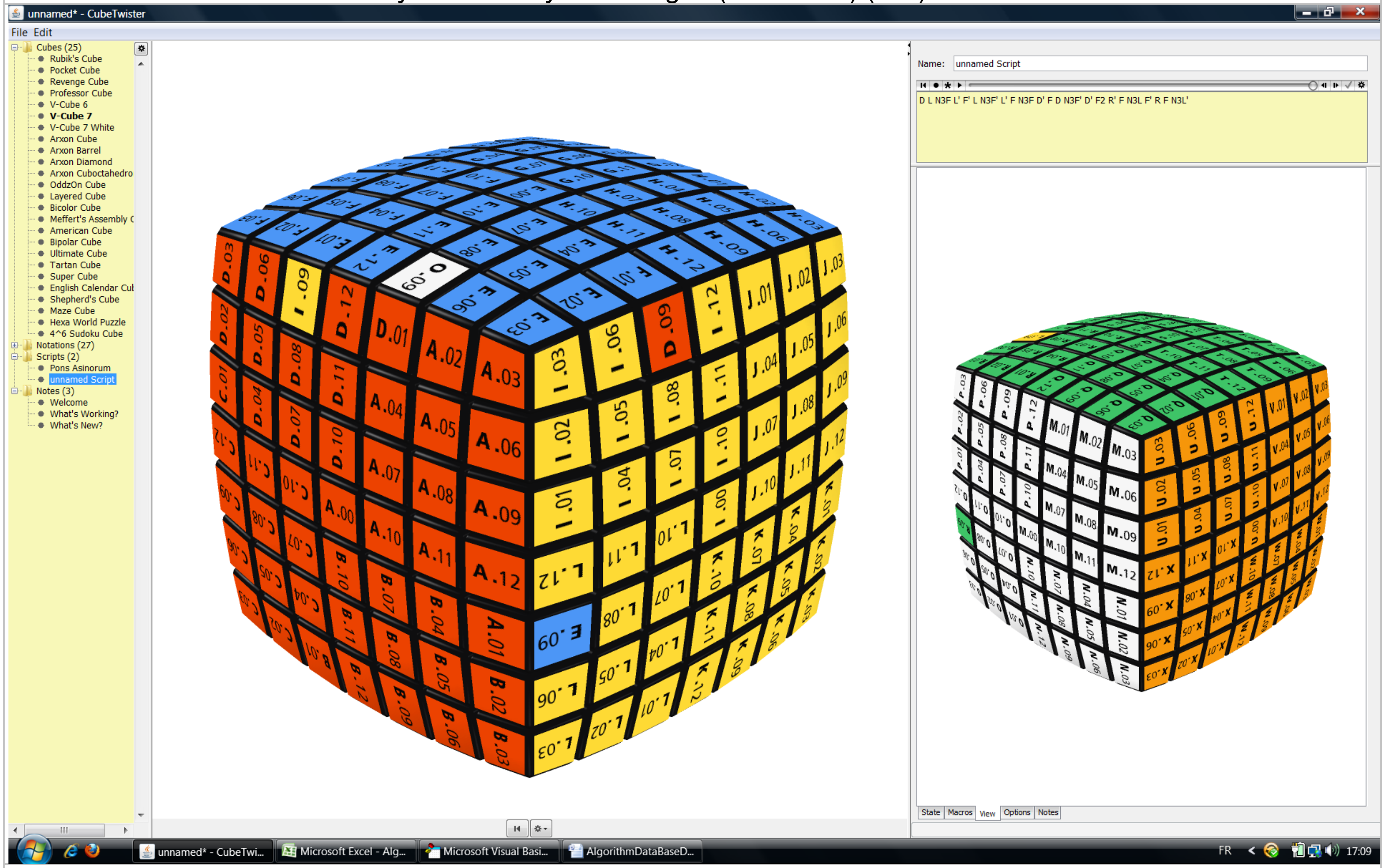
Corner Twists: [✓ Corner 1: CW - Corner 2: CCW] Corner 1: CCW - Corner 2: CW

Moves: 3-cycle, 5-cycle, 7-cycle, 9-cycle, 11-cycle, 2 2-cycles, [✓ 1 4-cycle + 1 2-cycle], Corner Twists, Midge Flips, Edge Parity Fix, Center Twists

Moves: 23 Algorithm: D L N3F L' F' L N3F' L' F N3F D' F D N3F' D' F2 R' F N3L F' R F N3L'

OK

1 4-Cycle + 1 2-Cycle of Edges (A B C D) (E F) – Orbit 01



unnamed* - CubeTwister

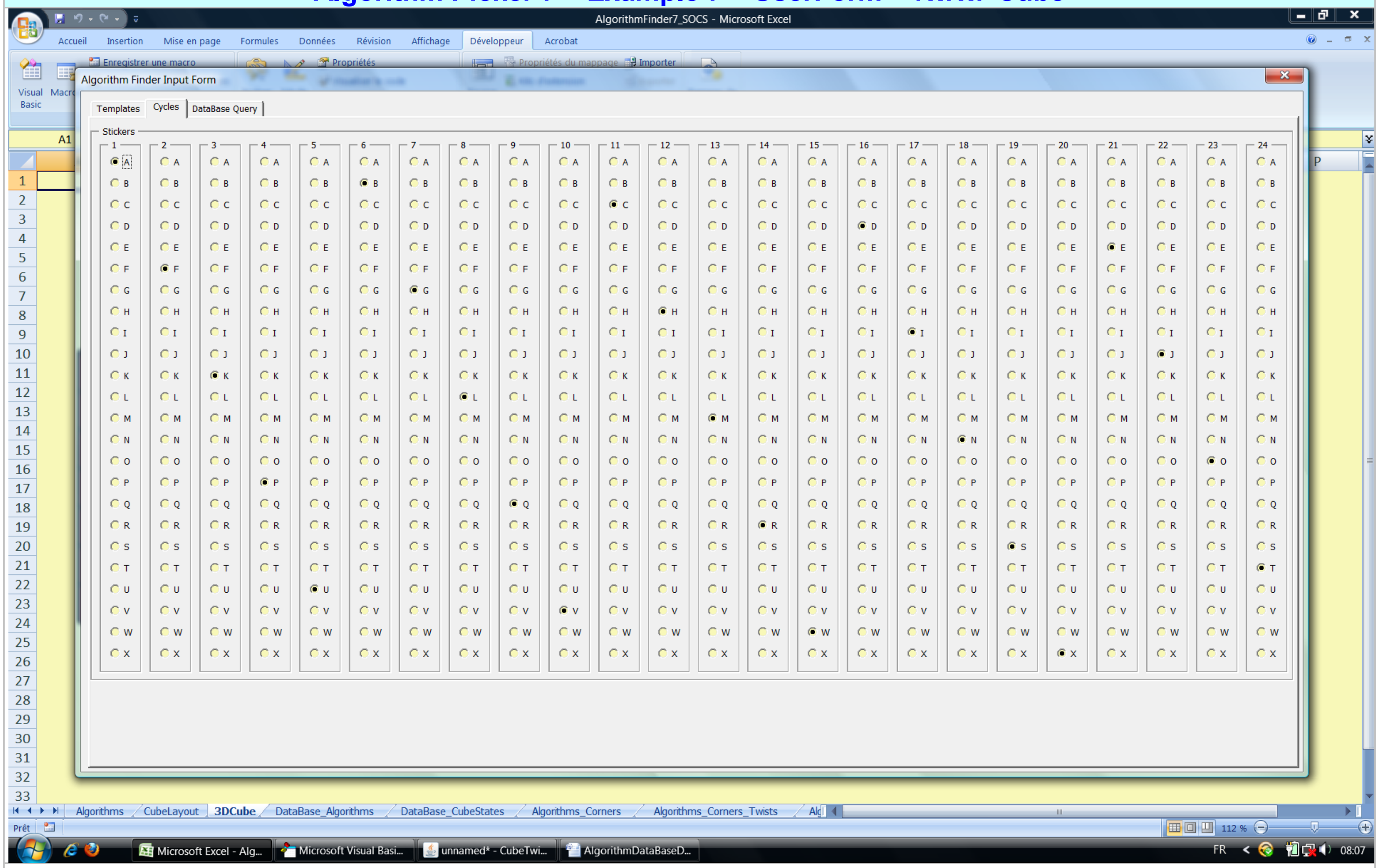
Name: unnamed Script

D L N3F L' F' L N3F' L' F N3F D' F D N3F' D' F2 R' F N3L F' R F N3L'

D L N3F L' F' L N3F' L' F N3F D' F D N3F' D' F2 R' F N3L F' R F N3L' (23 moves)

Algorithm Picker 7 – Example 7

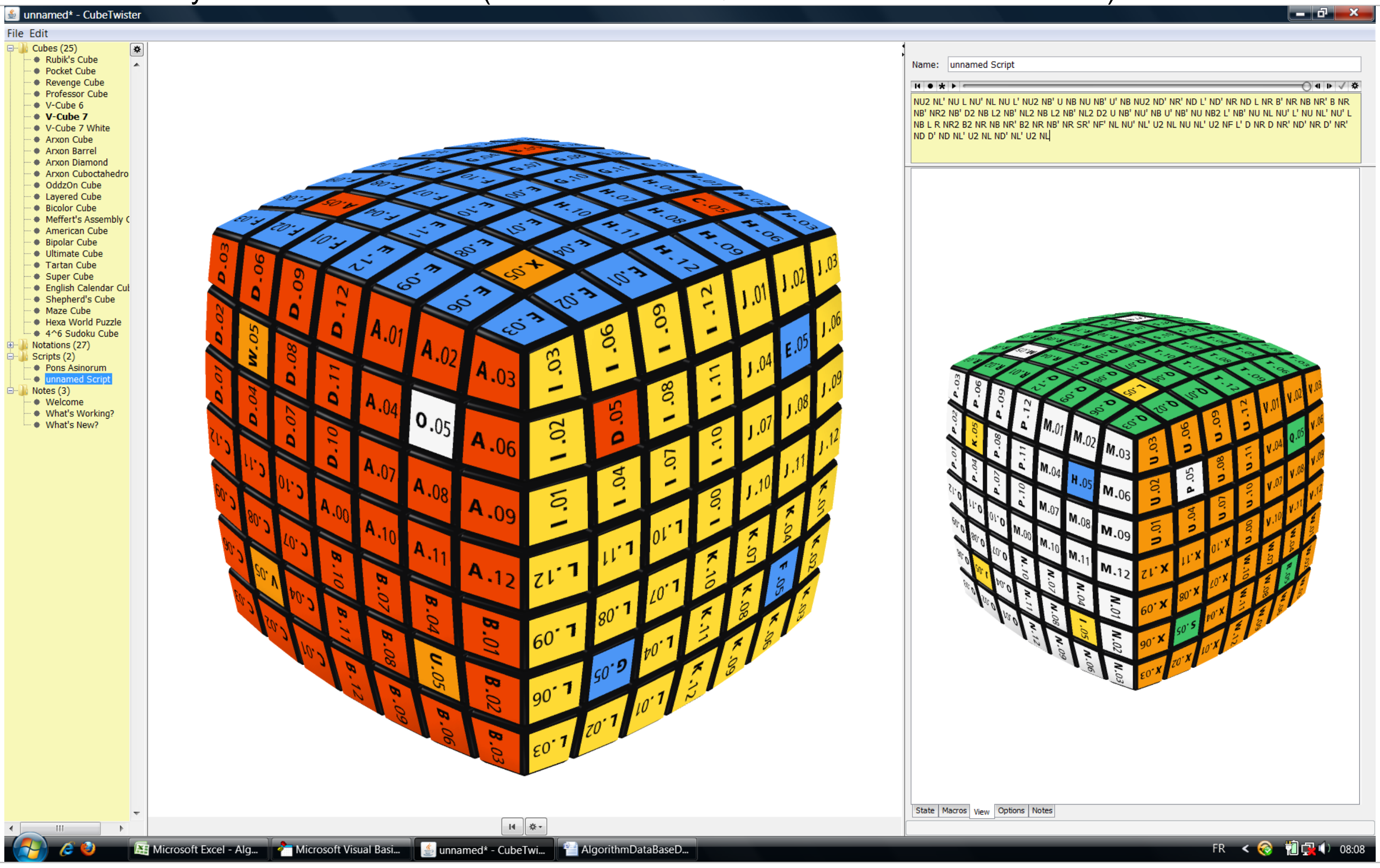
Algorithm Picker 7 – Example 7 – UserForm – 7x7x7 Cube



Stickers

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
[✓ A]	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
B	B	B	B	B	[✓ B]	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
C	C	C	C	C	C	C	C	C	C	[✓ C]	C	C	C	C	C	C	C	C	C	C	C	C	C
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	[✓ D]	D	D	D	D	D	D	D	D
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	[✓ E]	E	E	E
F	[✓ F]	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	[✓ G]	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H	H	[✓ H]	H	H	H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	[✓ I]	I	I	I	I	I	I	I
J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	[✓ J]	J	J
K	K	[✓ K]	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
L	L	L	L	L	L	L	[✓ L]	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
M	M	M	M	M	M	M	M	M	M	M	M	[✓ M]	M	M	M	M	M	M	M	M	M	M	M
N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	[✓ N]	N	N	N	N	N	N
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	[✓ O]	O
P	P	P	[✓ P]	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
Q	Q	Q	Q	Q	Q	Q	Q	[✓ Q]	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R	R	R	R	[✓ R]	R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	[✓ S]	S	S	S	S	S
T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	[✓ T]
U	U	U	U	[✓ U]	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
V	V	V	V	V	V	V	V	V	[✓ V]	V	V	V	V	V	V	V	V	V	V	V	V	V	V
W	W	W	W	W	W	W	W	W	W	W	W	W	W	[✓ W]	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	[✓ X]	X	X	X	X

23-Cycle of Corner-Centers (A F K P U B G L Q V C H M R W D I N S X E J O) – Orbit 05



Name: unnamed Script

NU2 NL' NU L NU' NL NU L' NU2 NB' U NB NU NB' U' NB NU2 ND' NR' ND L' ND' NR ND L NR B' NR NB NR' B NR NB' NR2 NB' D2 NB L2 NB' NL2 NB L2 NB' NL2 D2 U NB' NU' NB U' NB' NU NB2 L' NB' NU NL NU' L' NU NL' NU' L NB L R NR2 B2 NR NB NR' B2 NR NB' NR SR' NF' NL NU' NL' U2 NL NU NL' U2 NF L' D NR D NR' ND' NR D' NR' ND D' ND NL' U2 NL ND' NL' U2 NL

NU2 NL' NU L NU' NL NU L' NU2 NB' U NB NU NB' U' NB NU2 ND' NR' ND L' ND' NR ND L NR B' NR NB NR' B NR NB' NR2 NB' D2 NB L2 NB' NL2 NB L2 NB' NL2 D2 U NB' NU' NB U' NB' NU NB2 L' NB' NU NL NU' L' NU NL' NU' L NB L R NR2 B2 NR NB NR' B2 NR NB' NR SR' NF' NL NU' NL' U2 NL NU NL' U2 NF L' D NR D NR' ND' NR D' NR' ND D' ND NL' U2 NL ND' NL' U2 NL (105 moves)

Annex

Cube Numbering System

Sticker Numbering – 7x7x7 Cube

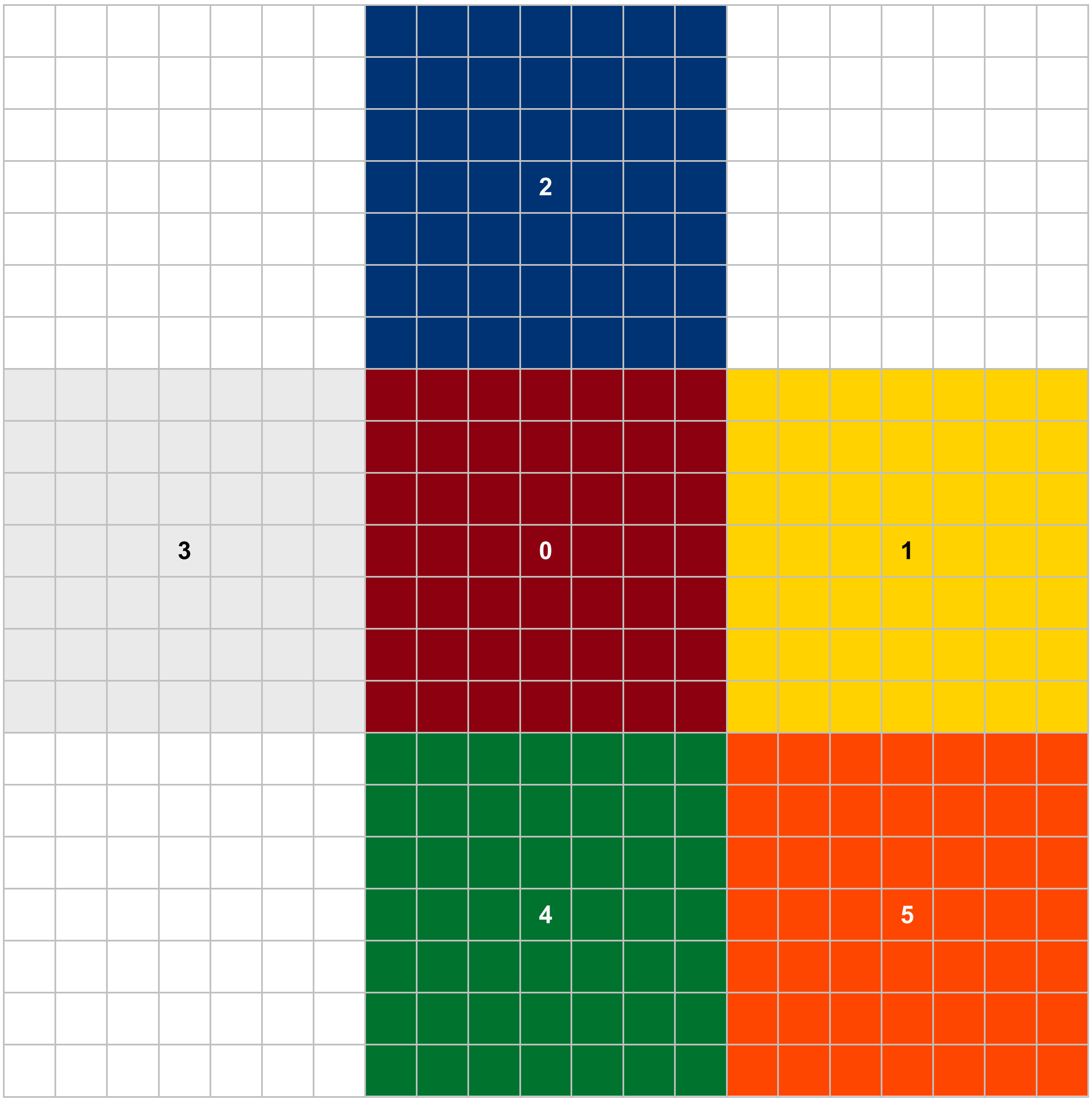
Stickers on a 7x7x7 cube are numbered *in rows* from 0 up to 293, beginning on face F, top left and ending on face B, bottom right, following sequence F R U L D B. Stickers can also be numbered *per orbit* from 0 up to 23.

							98	99	100	101	102	103	104							
							105	106	107	108	109	110	111							
							112	113	114	115	116	117	118							
							119	120	121	122	123	124	125							
							126	127	128	129	130	131	132							
							133	134	135	136	137	138	139							
							140	141	142	143	144	145	146							
147	148	149	150	151	152	153	0	1	2	3	4	5	6	49	50	51	52	53	54	55
154	155	156	157	158	159	160	7	8	9	10	11	12	13	56	57	58	59	60	61	62
161	162	163	164	165	166	167	14	15	16	17	18	19	20	63	64	65	66	67	68	69
168	169	170	171	172	173	174	21	22	23	24	25	26	27	70	71	72	73	74	75	76
175	176	177	178	179	180	181	28	29	30	31	32	33	34	77	78	79	80	81	82	83
182	183	184	185	186	187	188	35	36	37	38	39	40	41	84	85	86	87	88	89	90
189	190	191	192	193	194	195	42	43	44	45	46	47	48	91	92	93	94	95	96	97
							196	197	198	199	200	201	202	245	246	247	248	249	250	251
							203	204	205	206	207	208	209	252	253	254	255	256	257	258
							210	211	212	213	214	215	216	259	260	261	262	263	264	265
							217	218	219	220	221	222	223	266	267	268	269	270	271	272
							224	225	226	227	228	229	230	273	274	275	276	277	278	279
							231	232	233	234	235	236	237	280	281	282	283	284	285	286
							238	239	240	241	242	243	244	287	288	289	290	291	292	293

	Up		
Left	Front	Right	Back
	Down	Back	

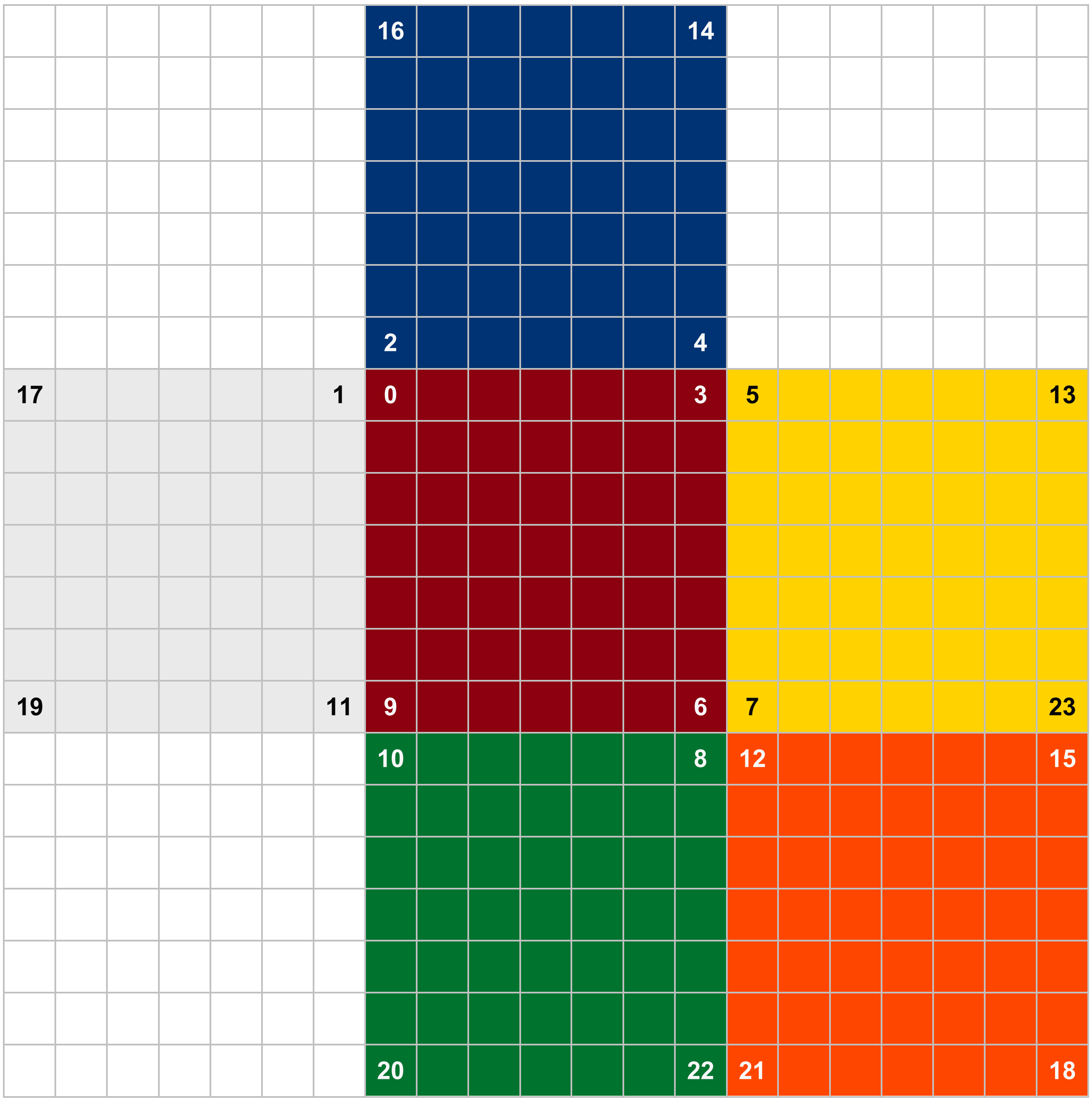
True Center Orbit

The 6 numbered stickers of the orbit of true centers of a 7x7x7 cube are shown on the texture below:



Corner Orbit – Stickers

The 24 numbered *stickers* of the corner orbit of a 7x7x7 cube are shown on the texture below:



Reference Cube								
Corner	0	1	2	3	4	5	6	7
Stickers	(0, 1, 2)	(3, 4, 5)	(6, 7, 8)	(9, 10, 11)	(12, 13, 14)	(15, 16, 17)	(18, 19, 20)	(21, 22, 23)
Orbit Cube								
Stickers	(D, O, F)	(A, E, I)	(B, L, S)	(C, R, P)	(W, J, H)	(X, G, N)	(U, M, Q)	(V, T, K)

Corner *stickers* are numbered from 0 to 23 whereas corner *pieces* are numbered from 0 to 7.

Sticker numbers have been selected such that a cyclic permutation of numbers in a group of 3 will always twist a corner in a *clockwise* direction, so that sticker permutation (0 → 1 → 2 → 0) will twist corner 0 clockwise (CW) whereas sticker permutation (0 → 2 → 1 → 0) will twist it counterclockwise (CCW).

For a group of 3 stickers (s0, s1, s2) located on a *same* corner piece, the corner number can be obtained from:

corner number = Integer(s0 / 3), or Integer(s1 / 3), or Integer(s2 / 3)

Given a corner sticker number s0, the 2 other corner sticker numbers can be obtained from:

s0 = 3 * Integer (s0 / 3) + (s0 + 0) Modulo 3

```
s1 = 3 * Integer (s0 / 3) + (s0 + 1) Modulo 3
s2 = 3 * Integer (s0 / 3) + (s0 + 2) Modulo 3
```

Corner Twists

There are four permuted corners per face turn. Quarter-turns turns F, F', B, B' and all half-turns F2, R2, U2, L2, D2, B2 do not change the twist of any of the four permuted corners, whereas quarter-turns R, R', U, U', L, L', D, D' change the twist of only *half* of the permuted corners.

To obtain the new twist of a corner that has been moved to a new position, simply add the delta twist value indicated in the Table of Corner Twists to the old corner twist:

newTwist = (oldTwist + deltaTwist) Modulo 3

For example, by applying move R to the right face, the new twist of the corner that has been moved from old position 4 to new position 7 is given by:

deltaTwist = +1
newTwist(7) = (oldTwist(4) + 1) Modulo 3

If this corner located at position 7 is now moved back to position 4 by move R' , then the 2 delta twists will cancel out Modulo 3:

deltaTwist = +2
newTwist(4) = (oldTwist(7) + 2) Modulo 3
newTwist(4) = ((oldTwist(4) + 1) + 2) Modulo 3
newTwist(4) = (oldTwist(4) + (1 + 2)) Modulo 3
newTwist(4) = (oldTwist(4) + 3) Modulo 3
newTwist(4) = oldTwist(4)

The table shows that any *legal* face move will *always* keep the sum of *all* corner twists equal to zero Modulo 3.

Table of Corner Twists								
Face Move F								
old corner position	0	1	2	3	4	5	6	7
new corner position	1	2	3	0	4	5	6	7
deltaTwist	+0	+0	+0	+0	+0	+0	+0	+0
Face Move F2								
old corner position	0	1	2	3	4	5	6	7
new corner position	2	3	0	1	4	5	6	7
deltaTwist	+0	+0	+0	+0	+0	+0	+0	+0
Face Move F'								
old corner position	0	1	2	3	4	5	6	7
new corner position	3	0	1	2	4	5	6	7
deltaTwist	+0	+0	+0	+0	+0	+0	+0	+0
Face Move R								
old corner position	0	1	2	3	4	5	6	7
new corner position	0	4	1	3	7	5	6	2
deltaTwist	+0	+2	+1	+0	+1	+0	+0	+2
Face Move R2								
old corner position	0	1	2	3	4	5	6	7
new corner position	0	7	4	3	2	5	6	1
deltaTwist	+0	+0	+0	+0	+0	+0	+0	+0
Face Move R'								
old corner position	0	1	2	3	4	5	6	7
new corner position	0	2	7	3	1	5	6	4
deltaTwist	+0	+2	+1	+0	+1	+0	+0	+2

Face Move U								
old corner position	0	1	2	3	4	5	6	7
new corner position	5	0	2	3	1	4	6	7
deltaTwist	+2	+1	+0	+0	+2	+1	+0	+0
Face Move U2								
old corner position	0	1	2	3	4	5	6	7
new corner position	4	5	2	3	0	1	6	7
deltaTwist	+0	+0	+0	+0	+0	+0	+0	+0
Face Move U'								
old corner position	0	1	2	3	4	5	6	7
new corner position	1	4	2	3	5	0	6	7
deltaTwist	+1	+2	+0	+0	+2	+1	+0	+0

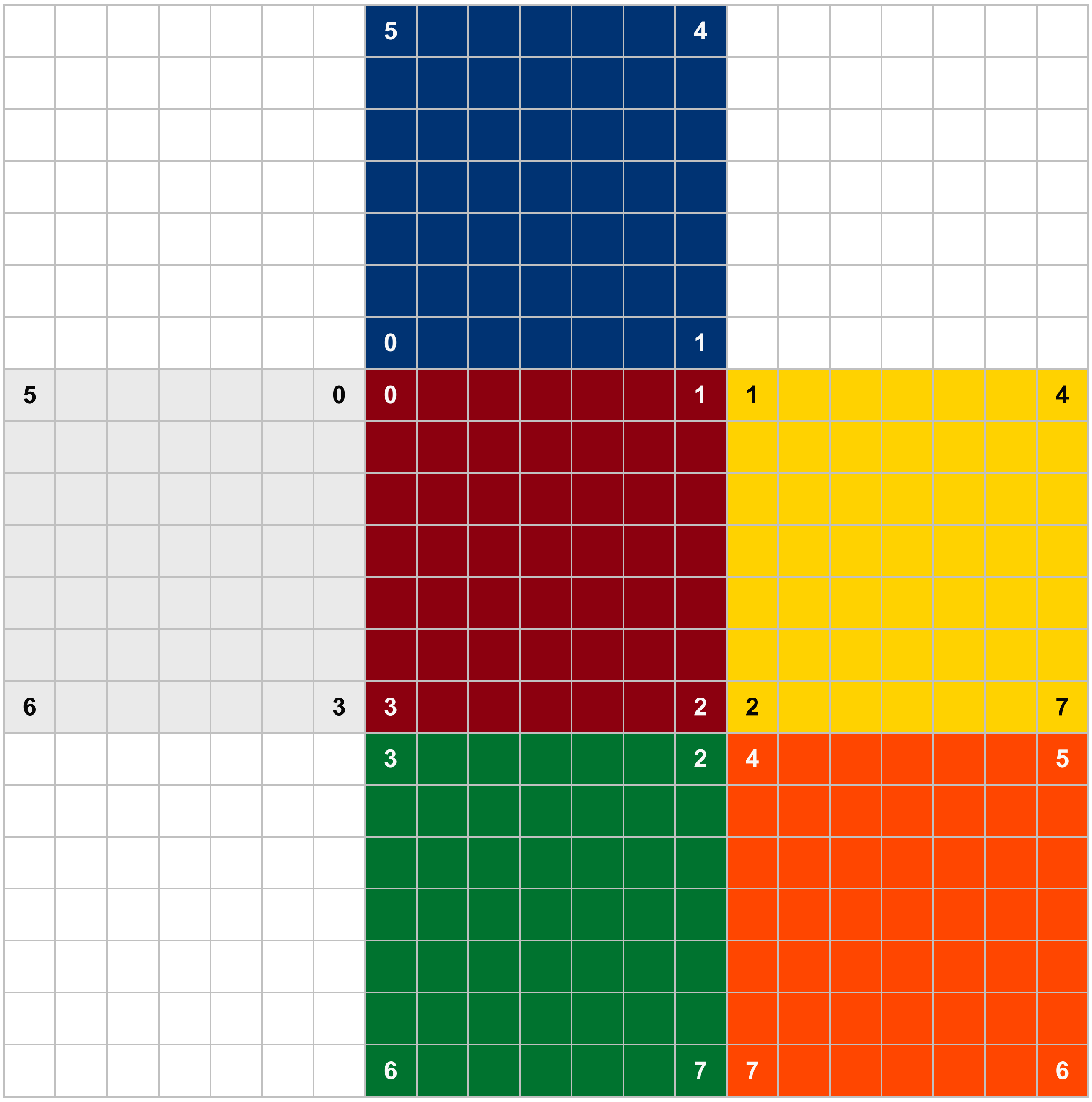
Face Move L								
old corner position	0	1	2	3	4	5	6	7
new corner position	3	1	2	6	4	0	5	7
deltaTwist	+1	+0	+0	+2	+0	+2	+1	+0
Face Move L2								
old corner position	0	1	2	3	4	5	6	7
new corner position	6	1	2	5	4	3	0	7
deltaTwist	+0	+0	+0	+0	+0	+0	+0	+0
Face Move L'								
old corner position	0	1	2	3	4	5	6	7
new corner position	5	1	2	0	4	6	3	7
deltaTwist	+1	+0	+0	+2	+0	+2	+1	+0

Face Move D								
old corner position	0	1	2	3	4	5	6	7
new corner position	0	1	7	2	4	5	3	6
deltaTwist	+0	+0	+2	+1	+0	+0	+2	+1
Face Move D2								
old corner position	0	1	2	3	4	5	6	7
new corner position	0	1	6	7	4	5	2	3
deltaTwist	+0	+0	+0	+0	+0	+0	+0	+0
Face Move D'								
old corner position	0	1	2	3	4	5	6	7
new corner position	0	1	3	6	4	5	7	2
deltaTwist	+0	+0	+2	+1	+0	+0	+2	+1

Face Move B								
old corner position	0	1	2	3	4	5	6	7
new corner position	0	1	2	3	5	6	7	4
deltaTwist	+0	+0	+0	+0	+0	+0	+0	+0
Face Move B2								
old corner position	0	1	2	3	4	5	6	7
new corner position	0	1	2	3	6	7	4	5
deltaTwist	+0	+0	+0	+0	+0	+0	+0	+0
Face Move B'								
old corner position	0	1	2	3	4	5	6	7
new corner position	0	1	2	3	7	4	5	6
deltaTwist	+0	+0	+0	+0	+0	+0	+0	+0

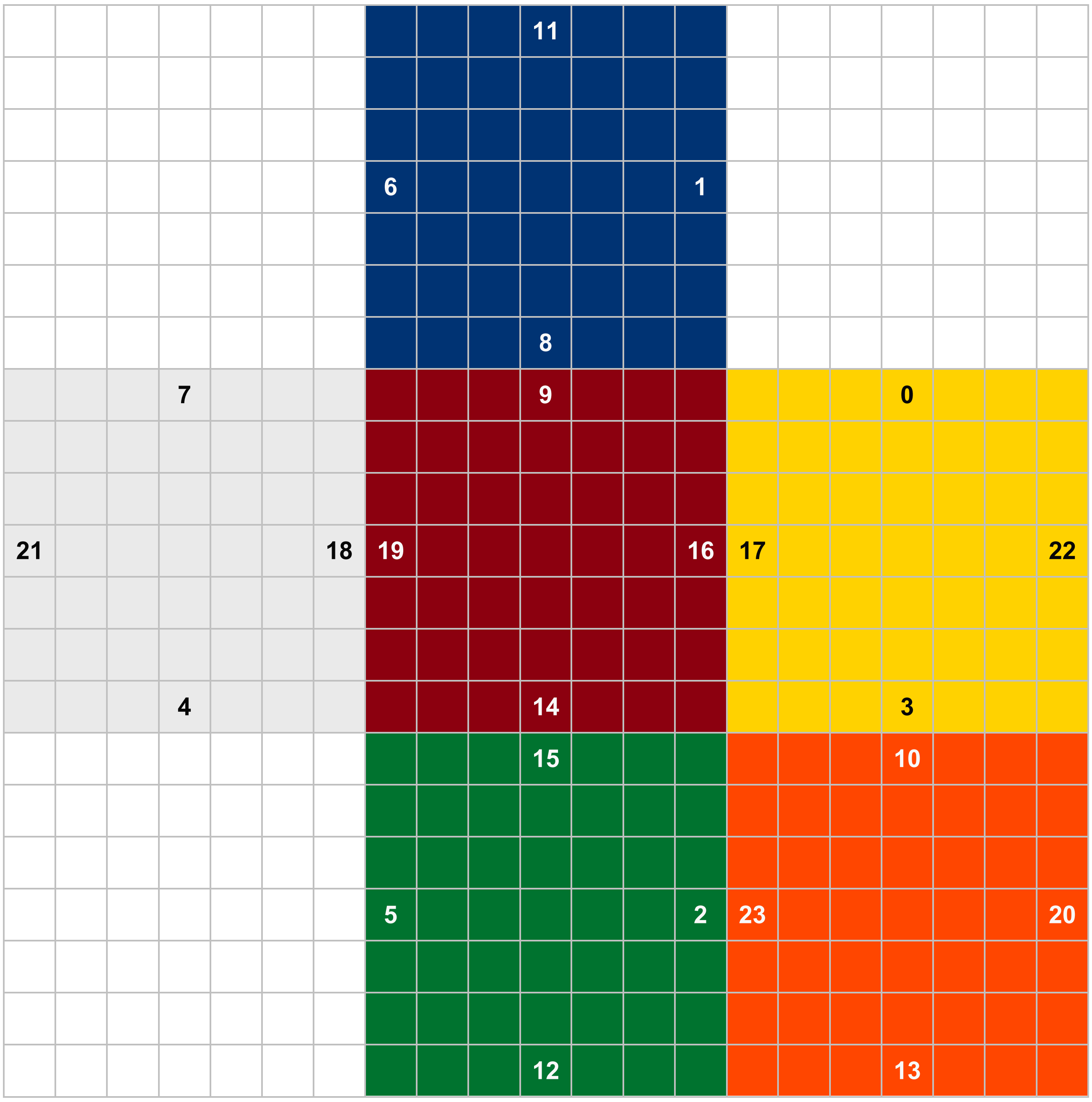
Corner Orbit – Pieces

The 8 numbered *pieces* of the corner orbit of a 7x7x7 cube are shown on the texture below:



Midge Orbit – Stickers

The 24 numbered *stickers* of the midge orbit of a 7x7x7 cube are shown on the texture below:



Reference Cube												
Midge	0	1	2	3	4	5	6	7	8	9	10	11
Stickers	(0, 1)	(2, 3)	(4, 5)	(6, 7)	(8, 9)	(10, 11)	(12, 13)	(14, 15)	(16, 17)	(18, 19)	(20, 21)	(22, 23)
Orbit Cube												
Stickers	(I, H)	(S, K)	(P, Q)	(F, N)	(E, D)	(W, G)	(T, U)	(B, R)	(A, L)	(O, C)	(X, M)	(J, V)

Midge *stickers* are numbered from 0 to 23 whereas midge *pieces* are numbered from 0 to 11.

For a pair of 2 stickers (s0, s1) located on a *same* midge piece, the midge piece number can be obtained as:

midge piece number = Integer(s0 / 2) or Integer(s1 / 2)

Given a midge sticker number s0, the number s1 of the sticker located on the other side can be obtained from:

s0 = 2 * Integer(s0 / 2) + (s0 + 0) Modulo 2
s1 = 2 * Integer(s0 / 2) + (s0 + 1) Modulo 2

Given a midge sticker number s that has moved to sticker *position* p, the midge flip can be obtained from:

1- Compute S = s - 2 * Integer (s / 2)
2- Compute P = p - 2 * Integer (p / 2)
3- Compare S and P and compute midge flip from:

if (P = S) then Flip = 0, else Flip = 1

Midge Flips

There are four permuted midges per face turn. Half-turns F2, R2, U2, L2, D2, B2 toggle the flip of each of the four permuted midges, whereas quarter-turns F, F', R, R', U, U', L, L', D, D', B, B' toggle the flip of only *half* of the permuted midges.

To obtain the new flip of a midge that has been moved to a new position, simply add the delta flip value indicated in the Table of Midge Flips to the old midge flip:

newFlip = (oldFlip + deltaFlip) Modulo 2

For example, by applying move R to the right face, the new flip of the midge that has been moved from old position 8 to new position 0 is given by:

deltaFlip = +1
newFlip(0) = (oldFlip(8) + 1) Modulo 2

If this midge located at position 0 is now moved back to position 8 by move R' , then the 2 delta flips will cancel out Modulo 2:

deltaFlip = +1
newFlip(8) = (oldFlip(0) + 1) Modulo 2
newFlip(8) = ((oldFlip(8) + 1) + 1) Modulo 2
newFlip(8) = (oldFlip(8) + (1 + 1)) Modulo 2
newFlip(8) = (oldFlip(8) + 2) Modulo 2
newFlip(8) = oldFlip(8)

The table shows that any *legal* face move will *always* keep the sum of *all* midge flips equal to zero Modulo 2.

Table of Midge Flips												
Face Move F												
old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	1	2	3	8	5	6	9	7	4	10	11
deltaFlip	+0	+0	+0	+0	+1	+0	+0	+1	+0	+0	+0	+0
Face Move F2												
old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	1	2	3	7	5	6	4	9	8	10	11
deltaFlip	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1
Face Move F'												
old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	1	2	3	9	5	6	8	4	7	10	11
deltaFlip	+0	+0	+0	+0	+0	+0	+0	+0	+1	+1	+0	+0

Face Move R												
old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	11	8	2	3	4	5	6	7	0	9	10	1
deltaFlip	+0	+0	+0	+0	+0	+0	+0	+0	+1	+0	+0	+1
Face Move R2												
old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	1	0	2	3	4	5	6	7	11	9	10	8
deltaFlip	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1
Face Move R'												
old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	8	11	2	3	4	5	6	7	1	9	10	0
deltaFlip	+1	+1	+0	+0	+0	+0	+0	+0	+0	+0	+0	+0

Face Move U

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	4	1	2	5	3	0	6	7	8	9	10	11
deltaFlip	+1	+0	+0	+1	+0	+0	+0	+0	+0	+0	+0	+0

Face Move U2

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	3	1	2	0	5	4	6	7	8	9	10	11
deltaFlip	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1

Face Move U'

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	5	1	2	4	0	3	6	7	8	9	10	11
deltaFlip	+0	+0	+0	+0	+1	+1	+0	+0	+0	+0	+0	+0

Face Move L

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	1	10	9	4	5	6	7	8	2	3	11
deltaFlip	+0	+0	+1	+1	+0	+0	+0	+0	+0	+0	+0	+0

Face Move L2

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	1	3	2	4	5	6	7	8	10	9	11
deltaFlip	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1

Face Move L'

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	1	9	10	4	5	6	7	8	3	2	11
deltaFlip	+0	+0	+0	+0	+0	+0	+0	+0	+0	+1	+1	+0

Face Move D

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	6	7	3	4	5	2	1	8	9	10	11
deltaFlip	+0	+0	+0	+0	+0	+0	+1	+1	+0	+0	+0	+0

Face Move D2

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	2	1	3	4	5	7	6	8	9	10	11
deltaFlip	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1

Face Move D'

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	7	6	3	4	5	1	2	8	9	10	11
deltaFlip	+0	+1	+1	+0	+0	+0	+0	+0	+0	+0	+0	+0

Face Move B

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	1	2	3	4	10	11	7	8	9	6	5
deltaFlip	+0	+0	+0	+0	+0	+0	+0	+0	+0	+0	+1	+1

Face Move B2

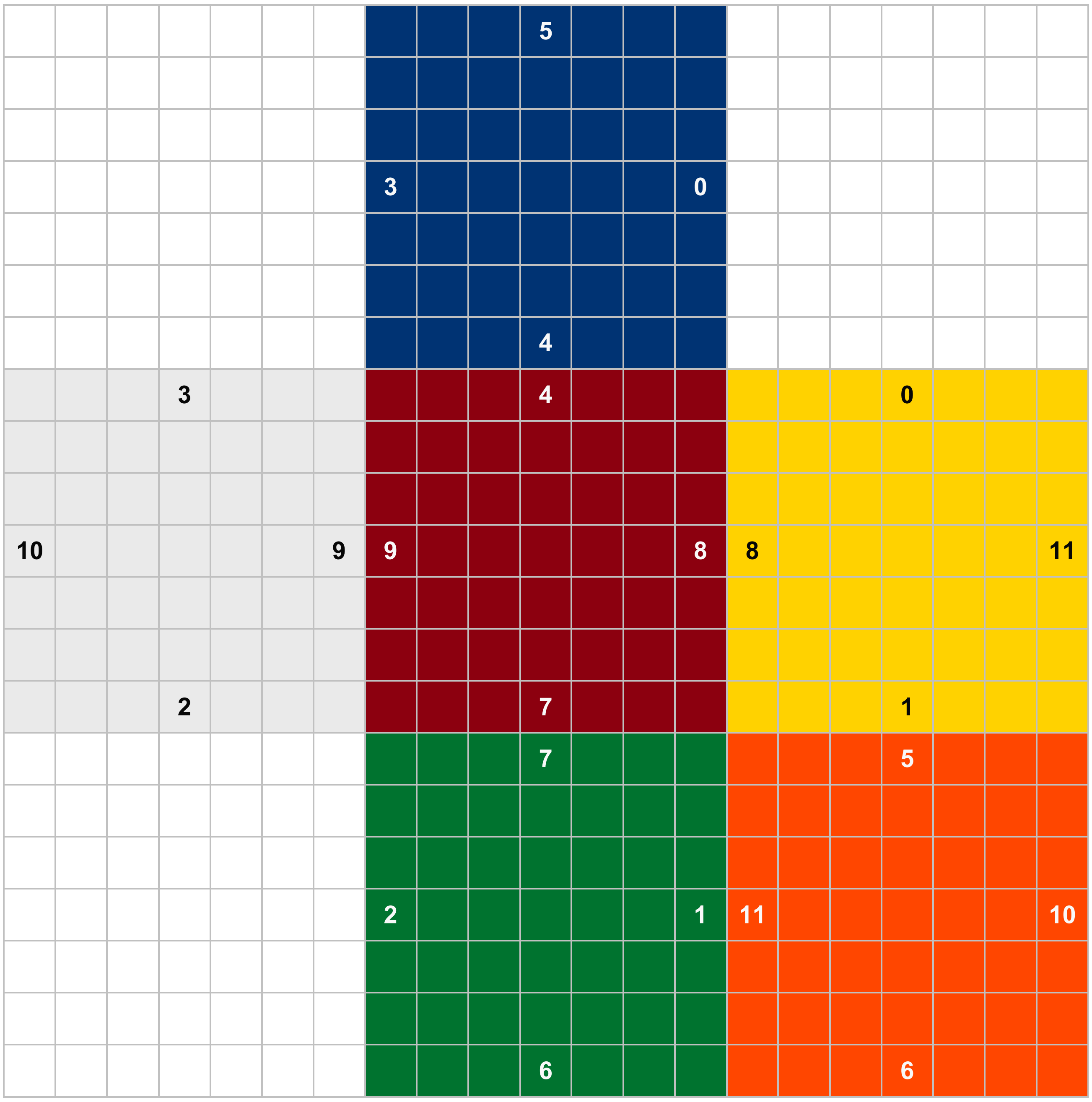
old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	1	2	3	4	6	5	7	8	9	11	10
deltaFlip	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1

Face Move B'

old midge position	0	1	2	3	4	5	6	7	8	9	10	11
new midge position	0	1	2	3	4	11	10	7	8	9	5	6
deltaFlip	+0	+0	+0	+0	+0	+0	+0	+0	+0	+0	+1	+1

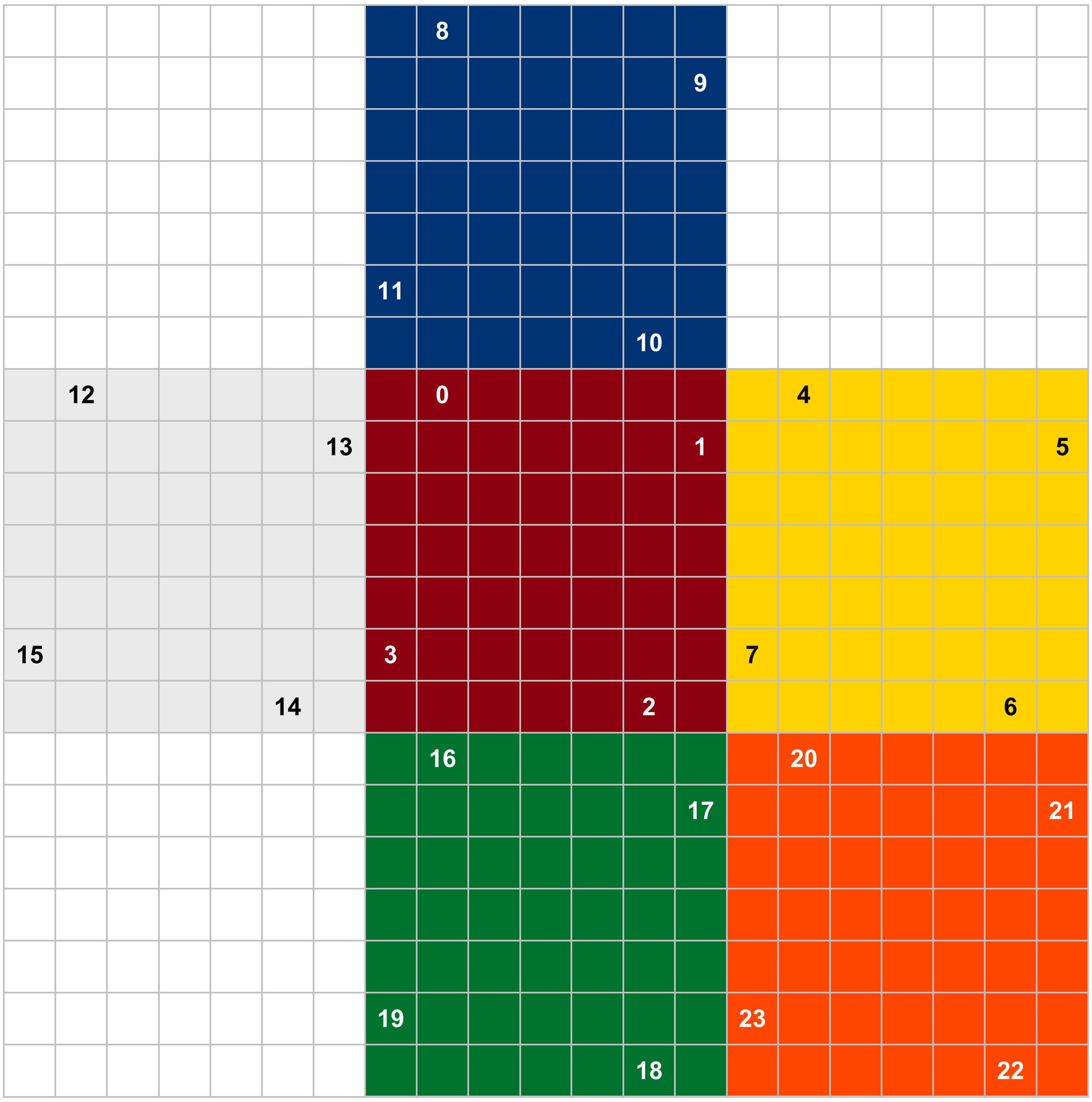
Midge Orbit – Pieces

The 12 numbered *pieces* of the midge orbit of a 7x7x7 cube are shown on the texture below:

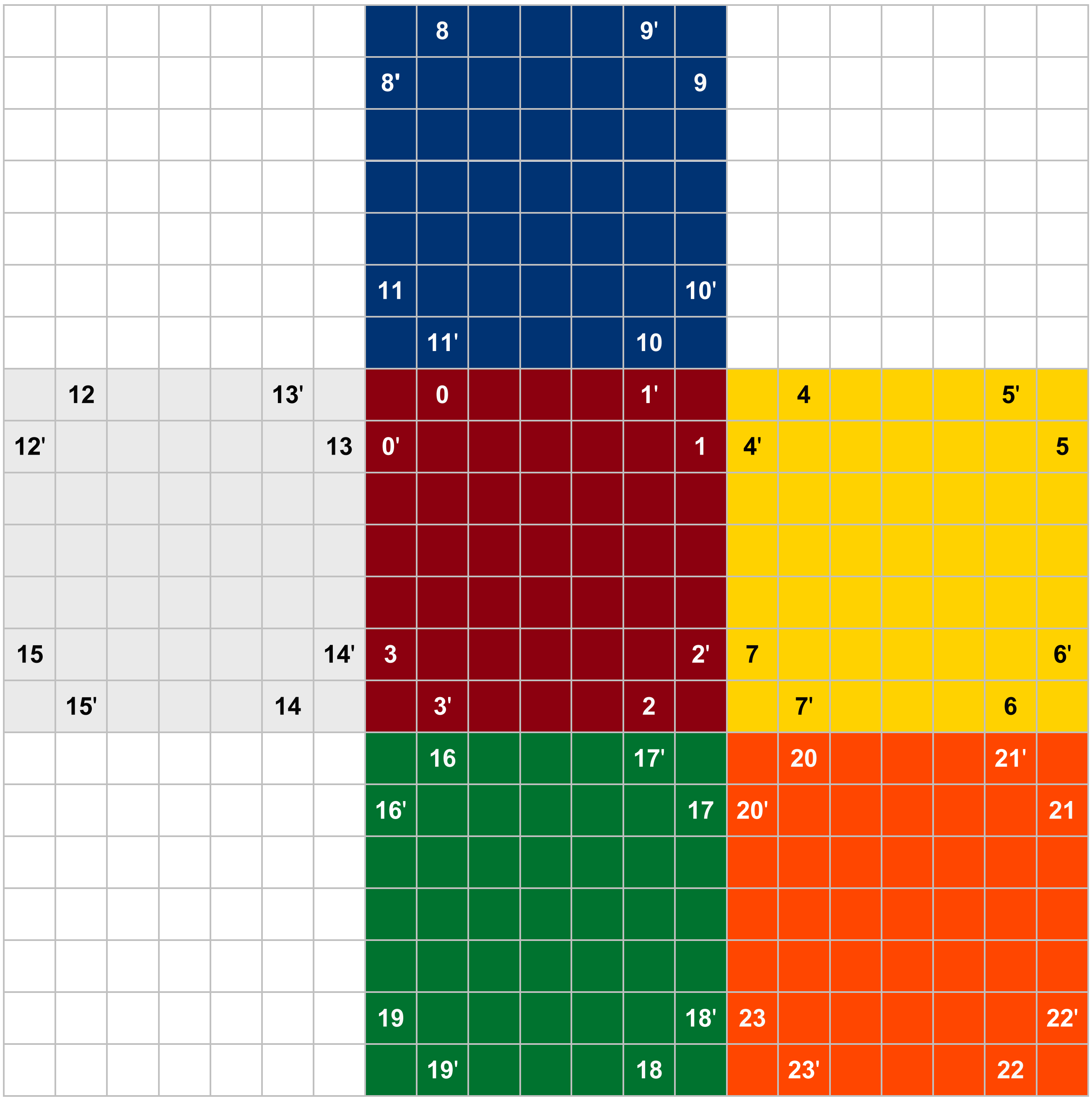


Edge Orbit 1

The 24 numbered *stickers* of the 1st edge orbit of a 7x7x7 cube are shown on the texture below (one edge side):



The numbered stickers of the 1st edge orbit of a 7x7x7 cube are shown on the texture below (two edge sides).

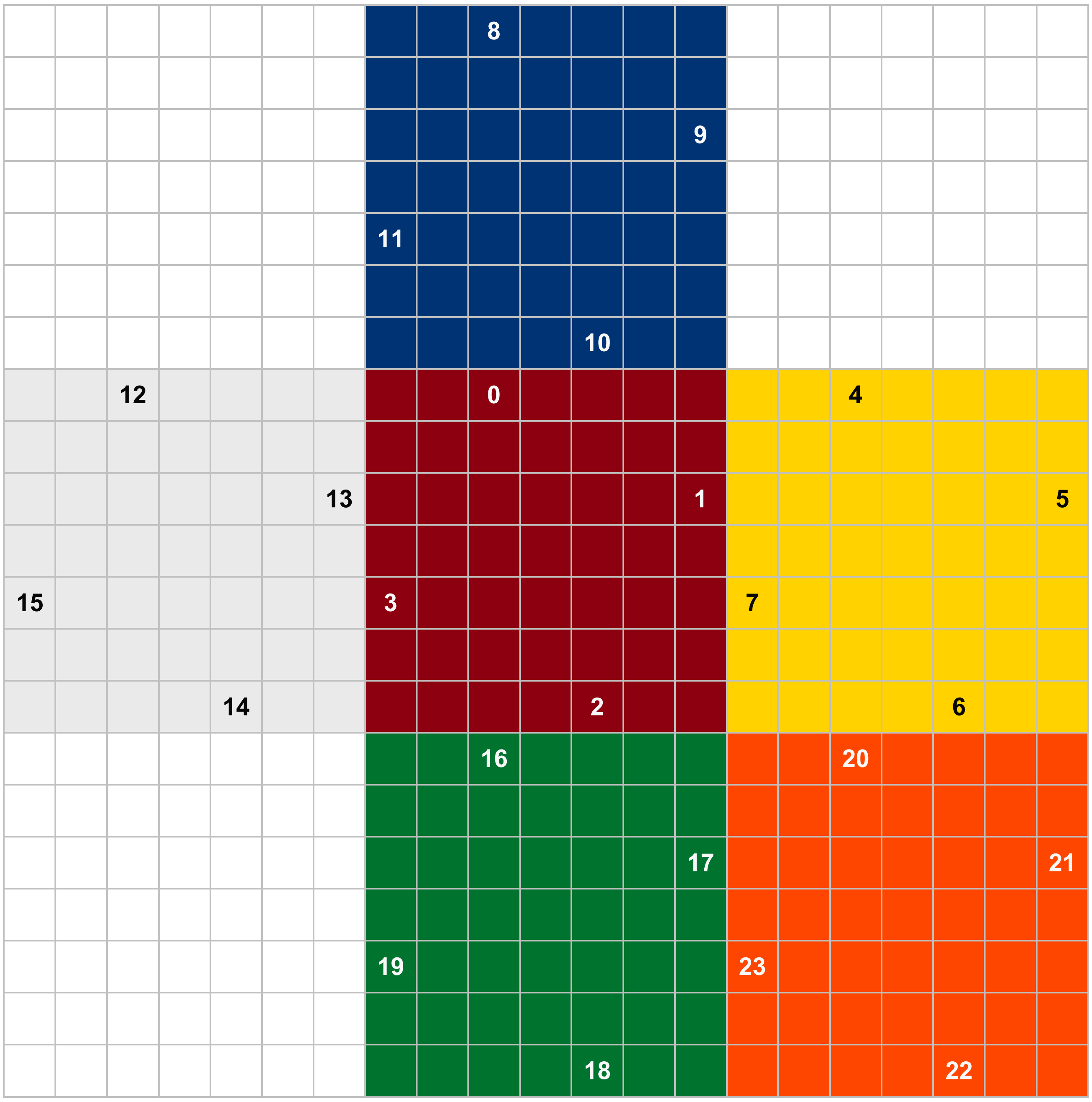


It can be seen that stickers of the two sides of an edge are paired. If an algorithm exists for cycling stickers of a side of an edge, then the same algorithm will also cycle stickers of the other side.

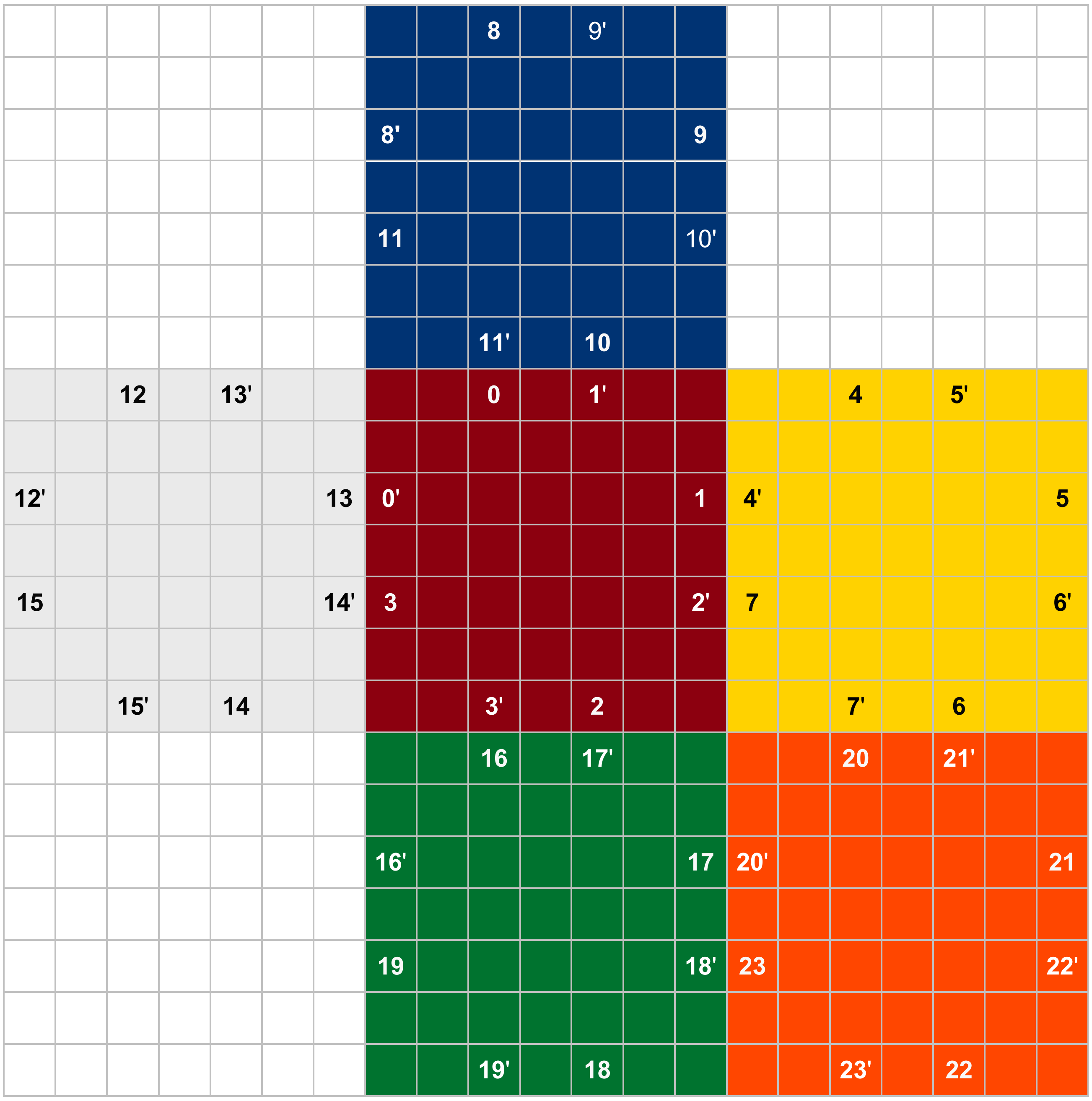
Edge Orbit 1 – Paired Stickers																							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
11'	4'	17'	14'	10'	20'	18'	2'	21'	5'	1'	13'	8'	0'	16'	22'	3'	7'	23'	15'	9'	12'	19'	6'
0'	1'	2'	3'	4'	5'	6'	7'	8'	9'	10'	11'	12'	13'	14'	15'	16'	17'	18'	19'	20'	21'	22'	23'
13	10	7	16	1	9	23	17	12	20	4	0	21	11	3	19	14	2	6	22	5	8	15	18

Edge Orbit 2

The 24 numbered stickers of the 2nd edge orbit of a 7x7x7 cube are shown on the texture below (one edge side):



The numbered stickers of the 2nd edge orbit of a 7x7x7 cube are shown on the texture below (two edge sides):

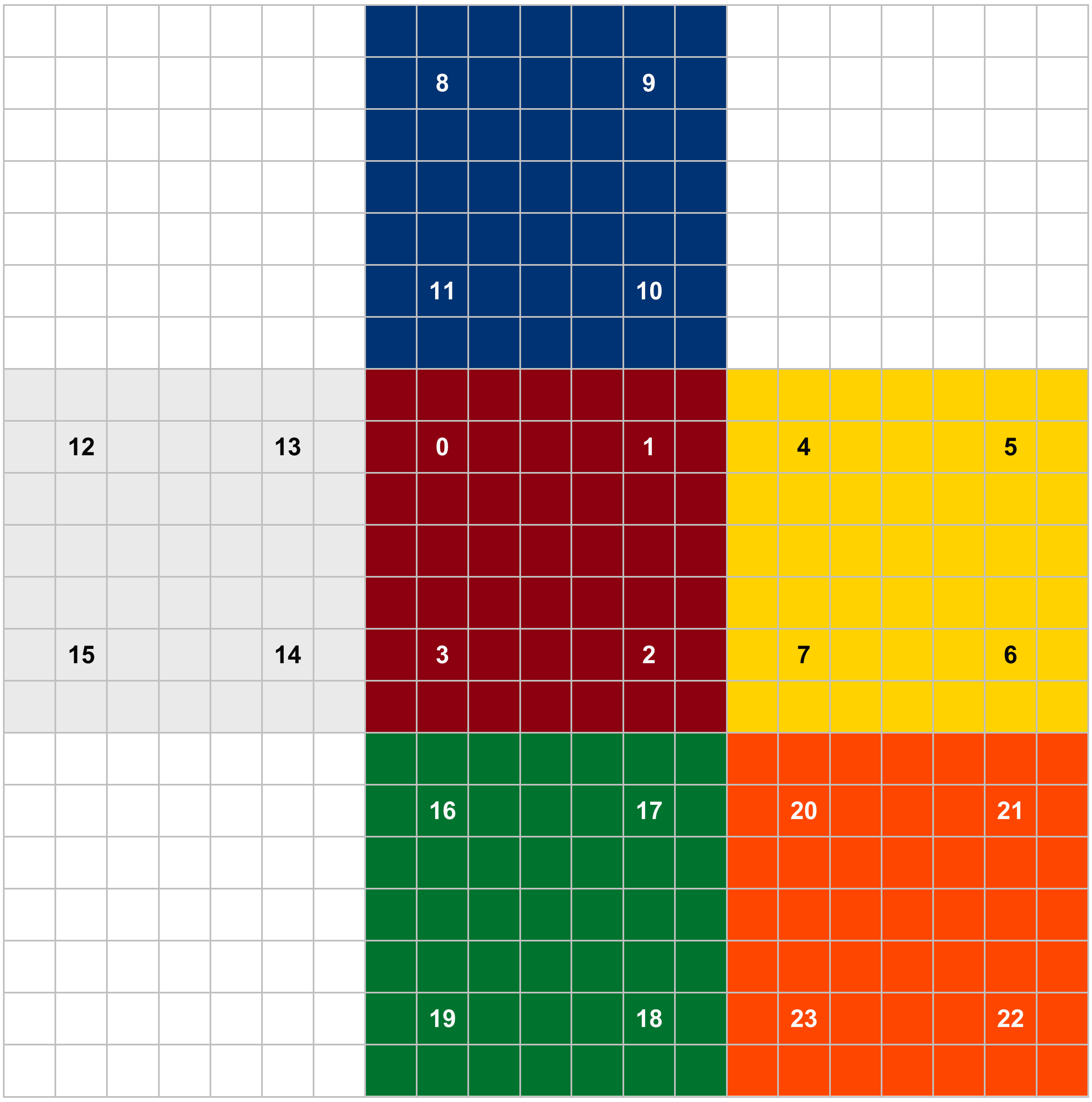


It can be seen that stickers of the two sides of an edge are paired. If an algorithm exists for cycling stickers of a side of an edge, then the same algorithm will also cycle stickers of the other side.

Edge Orbit 2 – Paired Stickers																							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
11'	4'	17'	14'	10'	20'	18'	2'	21'	5'	1'	13'	8'	0'	16'	22'	3'	7'	23'	15'	9'	12'	19'	6'
0'	1'	2'	3'	4'	5'	6'	7'	8'	9'	10'	11'	12'	13'	14'	15'	16'	17'	18'	19'	20'	21'	22'	23'
13	10	7	16	1	9	23	17	12	20	4	0	21	11	3	19	14	2	6	22	5	8	15	18

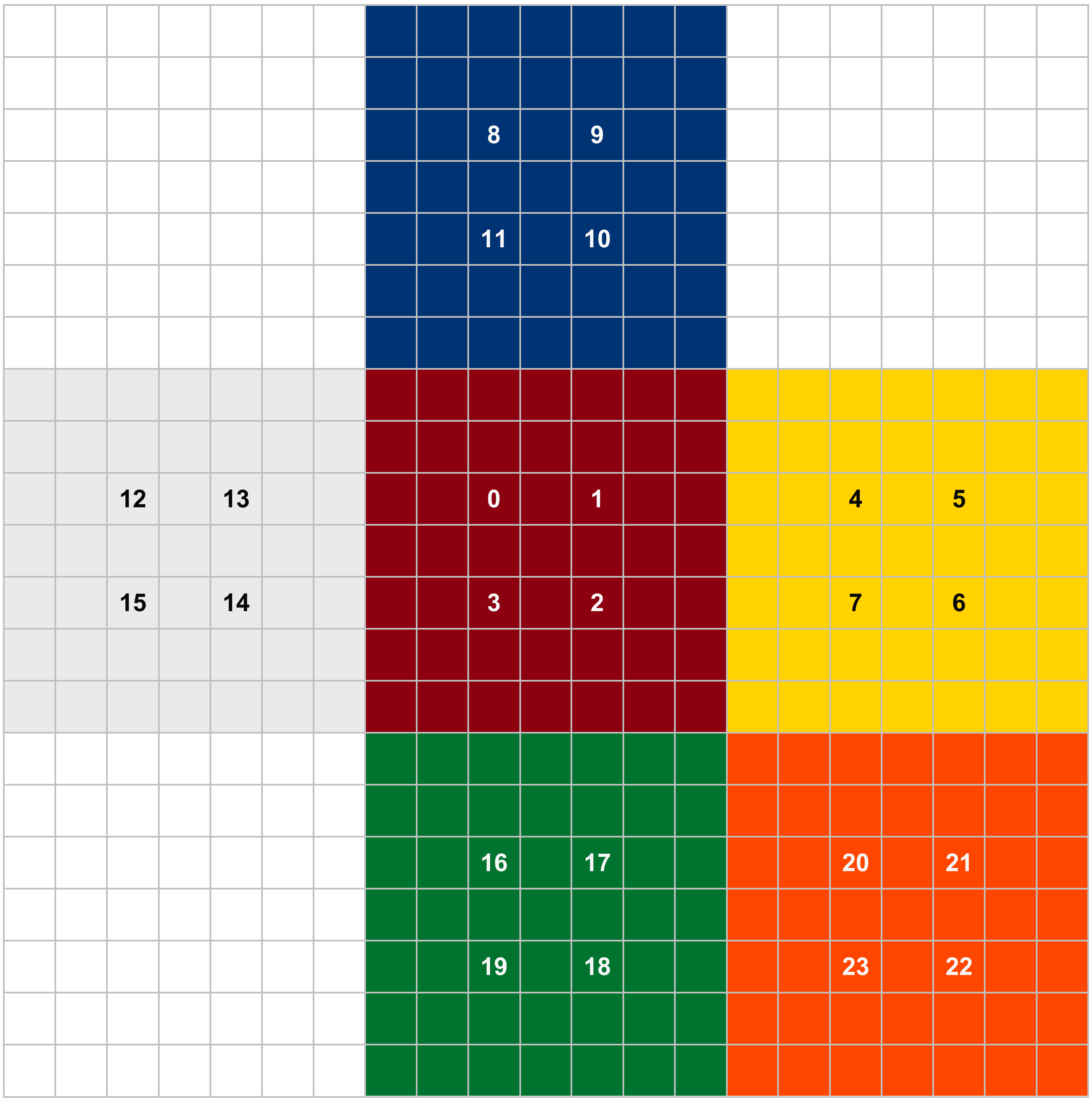
Corner-Center Orbit 1

The 24 numbered stickers of the 1st corner-center orbit of a 7x7x7 cube are shown on the texture below:



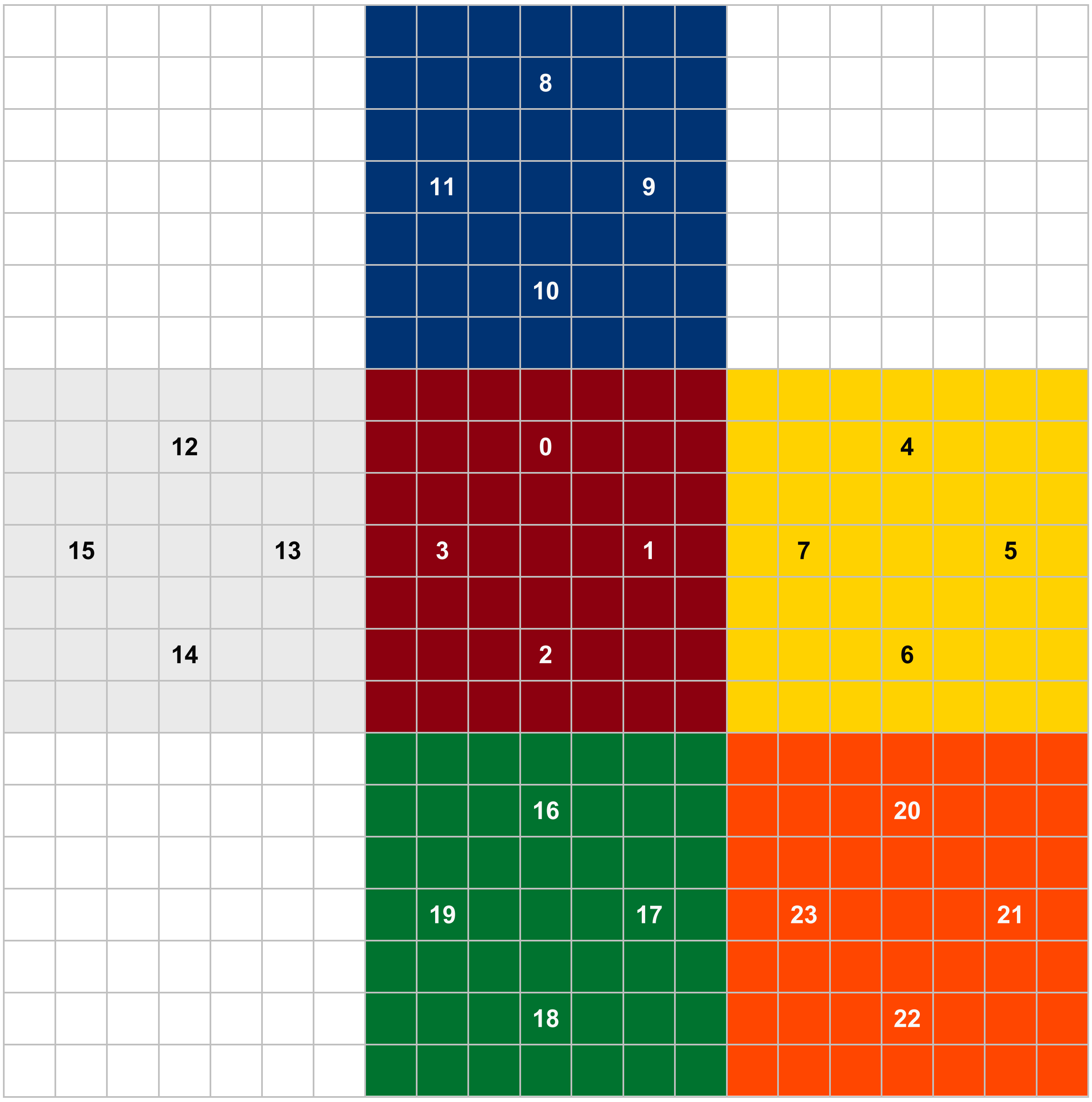
Corner-Center Orbit 2

The 24 numbered stickers of the 2nd corner-center orbit of a 7x7x7 cube are shown on the texture below:



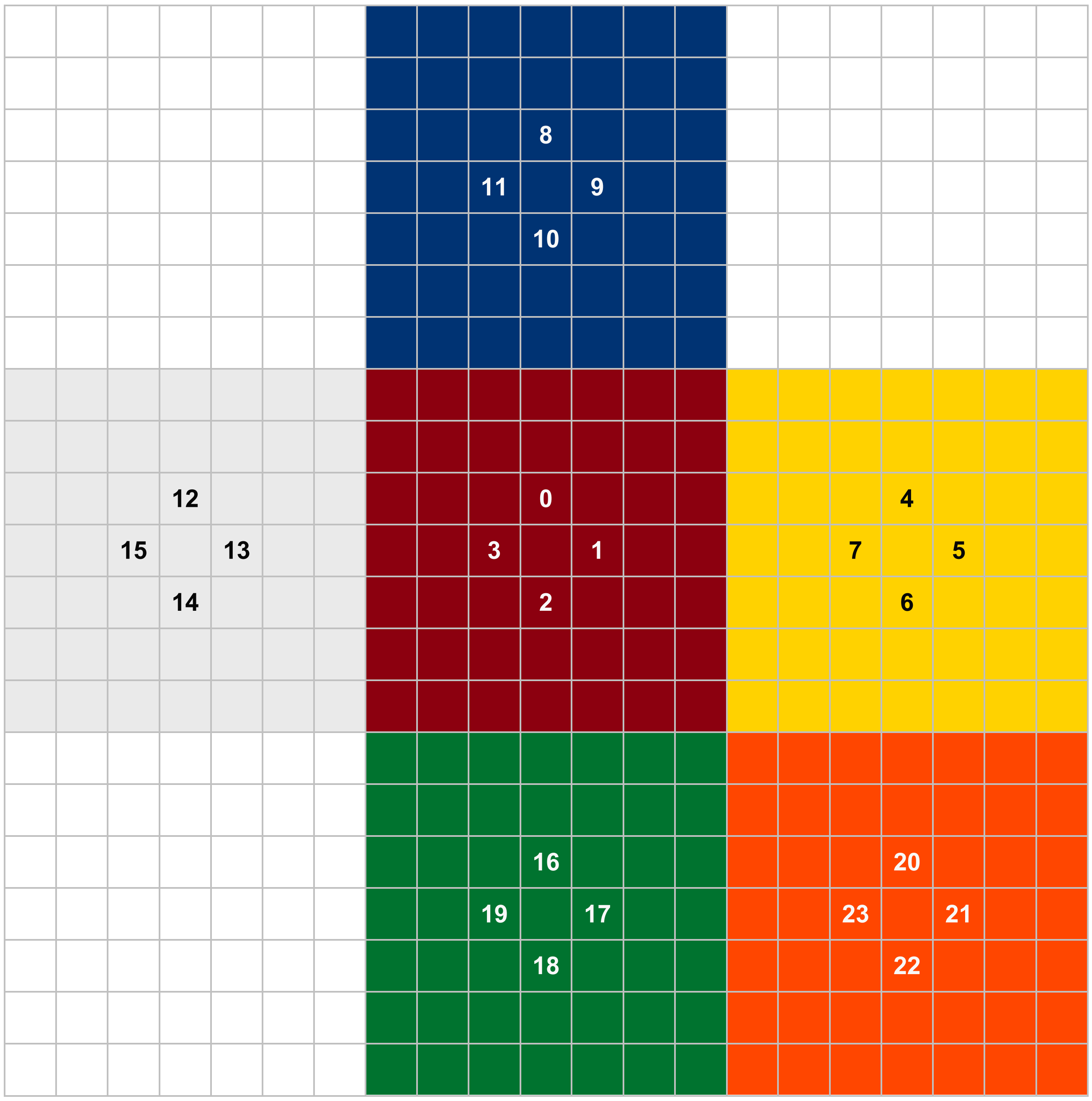
Midge-Center Orbit 1

The 24 numbered stickers of the 1st midge-center orbit of a 7x7x7 cube are shown on the texture below:



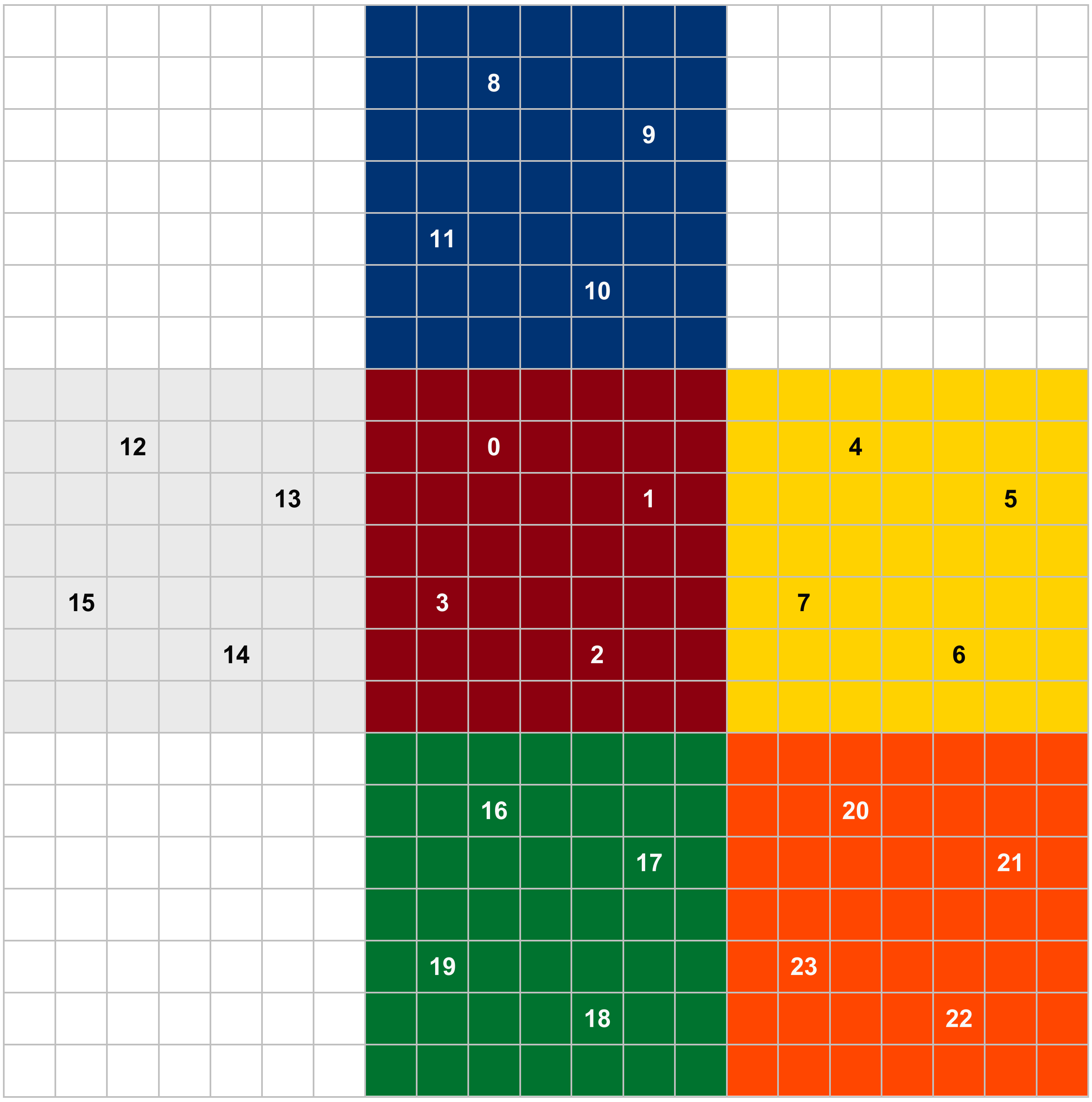
Midge-Center Orbit 2

The 24 numbered stickers of the 2nd midge-center orbit of a 7x7x7 cube are shown on the texture below:



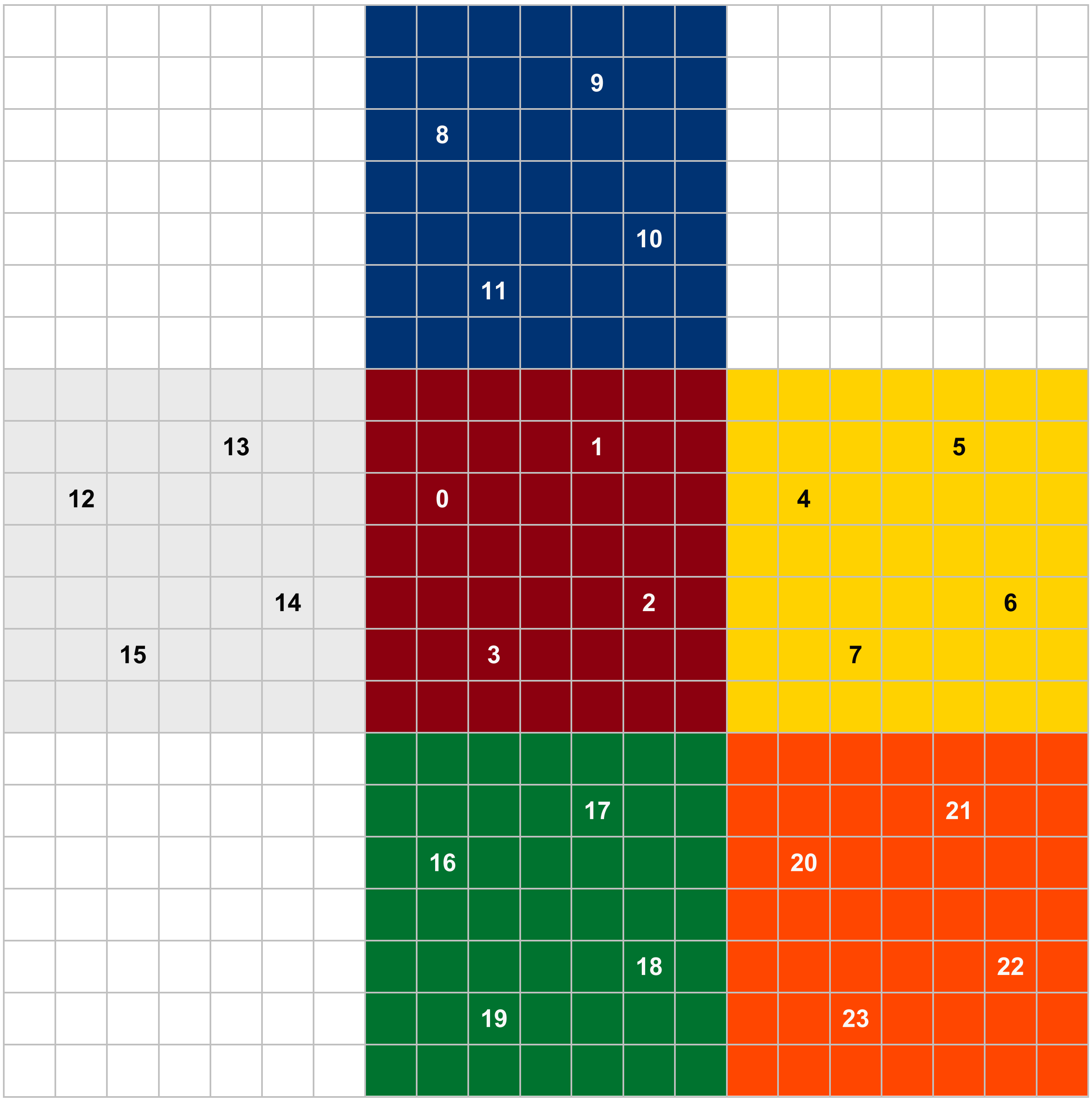
Edge-Center Orbit 1

The 24 numbered stickers of the 1st edge-center orbit of a 7x7x7 cube are shown on the texture below:



Edge-Center Orbit 2

The 24 numbered stickers of the 2nd edge-center orbit of a 7x7x7 cube are shown on the texture below:


9
8
10
11
13
1
5
12
0
4
14
2
6
15
3
7
17
21
16
20
18
22
19
23