Min-Steps

a Game Theory Algorithm

Author: Jeff Setterholm Lakeville MN, USA

January 28, 2025

The link to this paper: https://ftp.setterholm.com/WorldPeace/Math/Rubik/Min-Steps.pdf

This is a starting point for: "WisdomCAD"
... beneficial, extremely efficient decision making

"wisdom": *"Achieving harmonious results with no wasted effort and minimum mayhem."* - here , 2025

"guile" : "Deceitful cunning; craft and treachery." - Webster , 1956

Wisdom may begin to compete with guile,
which has been a daunting task for unaided human minds.

Introduction:

In 2025, Computer Aided Design (CAD) software systems of many types are an integral part of design work. As a simple example: hobbyists routinely "print" three-dimensional shapes of CAD mechanical models directly from virtual views on their personal PC screens. Within a specialty, CAD systems archive, analyze, coach, communicate, display, harmonize, standardize & track multiple aspects of a design.

The CAD paradigm for understanding specialties is far more powerful than any particular CAD implementation, faintly resembling how 'the matrix inversion paradigm' permeates computational applied mathematics in a vast variety of forms, regardless of the implementation details.

For at least hundreds of years "Wisdom" has been an almost-meaningless buzz word. This is a consequence of poorly conceived, self-serving & subjective "peer review" within academia. Hence everyone learns early-on that "Utopians are fools" …rather than peacefully-motivated optimists.

When CAD is unleashed on the "wisdom problem" dramatic advances will occur. 'Min-Steps', this paper, introduces a starting point for CAD system programmers to focus their rich, vibrant, analytical tradition on cooperative, extremely efficient human decision making. Being wise outside of academia will evolve from a "denied-by-snoots dream" down to an obvious, pleasant hobby option.

Demonstrating the Min-Steps algorithm:

A Rubik's Cube (`RC`) is solved when all the cells are properly oriented at the location where they belong. A 2x2x2 RC is now solved in the fewest number of moves (14 90° face rotations, or less) for all 3,6674,159 valid cube scrambles.

Solving the 2x2x2 cube and attempting to solve the 3x3x3 cube shape what follows.

Generic Variables: & as used here in Rubik's Cube solving:

Clear, concise definitions of key, short-named variables simplify computer codes & save brain-space.

Generic:	Rubik's Cube specific:	2x2x2	3x3x3	
A Attitude(s)	specific cell dispositions	21* of 24 each	24 each	="self-assessments"
C Choice(s)	specific face rotation option(s)	6	12	-without 180°'s
	- including 180°' rotations	9	18	
D Discord(s)	number of moves from solved	14 at most	unknown	-without 180°'s
E Emulator	model of physical cube behavior	2x2x2	3x3x3	
L Lookup table	the Discords of all scrambles	solved	not solved here	
M Move(s)	a face rotation(s) Choice			
R Result(s)	cube scramble(s)	total: 3,674,160	86.5033e18?≅ 86 quadrillion, maybe	
S Sequence(s)	set(s) of specific Moves' Choices			
V Voter(s)	any movable cube cell(s)	7* of 8	20 of 20	= "important entities"
W Workout	finding all novel Results in a Zone			
Z Zone(s)	Voter/Attitude sub-groups			
	with known Results Discords	*:the 2x2x2 has no reference axis,		
	- subset(s) of wisdom	so one cell with three possible		
		attitudes can remain fixed.		

Variable prefix & suffixes:

Prefix:

a_	"Address of"	every valid Result R has a unique addresses L(aR)
~	"approximately"	a subjective assertion, & my favorite symbol

Suffixes:

_i	"index of"	= [1,2,3,...,] Moves, Choices, Sequences, Voters, Attitudes & Results
_max	max number	
_n	novel	a previously unseen Result Rn
_p	the puzzle	a given scramble Rp
_s	solved	the solved cube Rs
_tot	total number	e.g.: M = [1,2,3,...,Mtot]
_'	"prime" =	a Result changed by a Move Choice
->	import/export	face rotations change scrambles (R & Ci) -> E -> R'

A Synopsis:

The Min-Step algorithm is a ~simple generic Workout that populates a Lookup table of Discord values within a Zone by generating Sequences for the Emulator and keeping track of Results novelty & hence Sequence novelty. Thus Min-Steps is able to pass a greatly-shortened stream of potentially-novel Sequences to the Emulator, and does so until no novel Results are returned.

The tricky part is realizing a Zone, which includes programming the Emulator with its input Sequence processing, exercising Move Choices yielding Voters Attitudes, and exporting of Results. The addresses of the Lookup table (e.g.: see page 8) are Zone-specific Voter Attitudes. When the Emulator no longer returns novel Results the Lookup table is complete. The RC Emulator is programmed herein a couple of different ways, both in a few lines of Fortran source code. Addresses are the **i**ndices of the Lookup table array; the "Indexer" that generates the Result address **aR** is much more complicated code, but has generic aspects.

Discords do not "count the votes", but rather reveal the downhill 'topology' (a multi-dimensional surface or surfaces) of the purely-cooperative Move Choices within a Zone. Zone/Discord use is wise planning's finest hour: with appropriate software ~effortlessly solve a Zone's puzzle in the absolute minimum number of moves. Beyond that, algorithms which achieve harmony across multiple Zones (mine don't, yet) will begin to automate "common sense". At best, elections are relatively blunt instruments for realizing constructive social change.

Overview:

As a veteran technical problem solver: I have gambled 21 months of my time that solving Rubik's Cubes in the fewest number of moves would enlighten my quest to figure out World Peace. And it has! But trying to solve scrambled cubes, per se, was a dead end. Likewise…

Figuring out World Peace is a seemingly overwhelming challenge. However, in conversations with other people, their expressible ideas suddenly abounded after the problem was rephrased:

"How could a perfect world be screwed into our present mess?"

Here, the entire downhill-to-solution topology of the 2x2x2 Rubik's Cube minimum-steps Move Choices is made accessible and becomes navigable after finding and cataloging all the cube scrambles by how-far-away-from-solved they are… by reverse engineering out to every last scramble from "solved". Know that your Emulators will be inside the ballpark when they can reverse engineer out to the existent screwed up Results of the problems that puzzle you.

A computer model of the Rubik's Cube, the Emulator **E**, has a central role. All the possible Cube face rotation Choices **C** are modeled to alter the location & rotation of each cell of the rotated face. It turns out that 24 cell Attitude values – **A**'s – also uniquely specify the cell locations.

Each cell is generically a Voter **V** with a known **A** value - its Vote. The Attitudes of all the Voters-of-interest is a Result **R**. RC Voters are divided into Zone(s) **Z** subset groups because considering them all at once for a 3x3x3 was too big a problem. The 8 corner cells of a 2x2x2 are a solved Zone that is fully functional.

E processes Sequences **S** of face rotation Move **M** Choices **C**, deconstructing the solved Result **Rs** to new scrambled Result **R'.** The Move count is called the Discord **D** of the Result when it is the smallest number of Moves that reach the Result.

Each **R'** has a unique address in a Lookup table **L**, and the Lookup table stores "how far from solved", i.e.: the Disord **D**, of each novel scramble **Rn**. **L**(**aRn**) = **D**. Initially the entire table is set to value **L**(1:**Rtot**) = -1. Because the Sequences are presented from the shortest (**D**=0) to the longest (**Dmax**=14, for the 6-Choice 2x2x2): if(**L**(**aR'**)=-1) then **R'** is novel & **L**(**aR'**) is set = D. Thus the Lookup table is populated with all the fewest Move D values and becomes ready for use.

The Lookup table is used by the Emulator to reveal the entire 2x2x2 downhill-to-solution topology. Any cube scramble is a puzzle **Rp**. All the Move choices are tested: each **C** yields (**Rp** & **C**) → **E** → **R'** and **L**(**aR'**) = **D** is the disorder. **Rp** has at least one **C** that will reduce **D** by one at the next **R'**, except for **Rs** at **D**=0, which can only move to a new **R'** with a **D**=1. And **L**(**aRp**)=**D** is the minimum number of 90° Moves that will be needed to solve **Rp**.

Hence, in the case of 2x2x2 Rubik's Cubes, knowing the Disorder of scrambles provides sufficient information to solve them in the fewest moves, without having to store a specific solution Sequence for each scramble. I.e.: Discords form a topology that transcends Move recipes.

For the RC solving problem, wisdom has very limited scope, with further division into Zones Z. Here, for an RC, solving/harmonizing seven corner cells (#1,#2,#3,#4,#5,#6,#7) of a 2x2x2 RC is Zone#2 (cell #8 is held fixed). For a 3x3x3, solving the four front-right-lower quad of cells (#8,#16,#19,#20) is Zone#1, and solving the nine non-quad edge cells (#9,#10,#11,#12,#13,#14,#15,#17,#18) is Zone#3. After solving Zone#1 then either Zone#2 corners or Zone#3 edges do solve in a minimum number of steps, but, for deep scrambles, not both at the same time. Perhaps there are bugs in my assumptions or my code. Nonetheless, Zone#2 alone does solve 2x2x2's.

Grouping Variables Generically: This rephrases the overview.

A Sequence has Moves consisting of Choices
Si **M** [1,2,3,,,,**Mtot**] **C** [C1,C2,C3,...**CMtot**]
Sequences are generated from the fewest Moves to the most moves

Results are a snapshot of the Voters` Attitudes.
Ri **V** [1,2,3,…,**Vtot**] **VA** [A1,A2,A3,...**AVtot**]

A Result and a Choice in The Emulator yield a new Result
(**R** & **C**) **->** **E** **->** **R'**
Some Voters' Attitudes are always modified in the Emulator by any Move Choice.

Use the Emulator repeatedly to produce a Sequence's deconstructed Result.
(**Rs** & **Si**) **->** **E** -> **Ri**

If **Ri** is novel **Rn**, i.e.: not seen before,
then the Disorder is the number of Moves of **Si**

Each Result has a unique Address in a Lookup table **L**
Initially all **L** values = -1
A Result is novel if the Lookup table address is negative.
If **L**(**a(Ri)**)<0 then set **L**(**a(Ri)** =**D**

Starting with no Moves i.e. **Rs** & **D**=0, exercise all of the Choices of the first move, and record the novel results for **D**=1, which will be all of them.

Repeat this process for the second move: test all the **D**=1 novel sequences using all the Choices, again saving the **D** values of the novel Results in the Lookup table. Not all the Results are novel, e.g.: reversing the **D**=1 move choice at **D**=2 returns the Result to solved.

Repeat this process for **D** = **D**+1. Eventually there will be no more Novel Results and the Lookup table is fully populated.

Generating & evaluating all the Sequences as described above is a Workout **W**.

In order to then use the Lookup table & Emulator as a real world puzzle solver, the **Rp** "problem" - a physically scrambled cube - needs to be imported into the software environment; tedious means to do that for a 2x2x2 are provided. Link #0L145 "ScrambleID-2x2x2.pdf".

Unusual aspects:

1. The **i**'s involved: **Si**, **Ri**, **aRi**, etc. can be huge.
2. Define Emulator variables as early as possible & as succinctly as possible.
3. Coding a robust Emulator was challenging; fast solutions save lots of time; therein concatenation (joining two functions so that their interface vanishes) is a gem;
4. Generating Sequences and Results using an existing Emulator is easy.
5. Computing the unique Lookup table addresses of Results can be difficult because the maximum address size can be too large for available computer memory.
6. Only using novel Results allows the Emulator to terminate a Workout sooner than expected.
7. Workout runtime and maximum address size are somewhat related.
8. For huge datasets, direct addressing Results via an Indexer is much faster than searching a sort-based list.

Reference .pdf files: This page is approximate, & the .zip files lag actual posts.

These seven .pdf files all have line numbers which are frequently referenced.

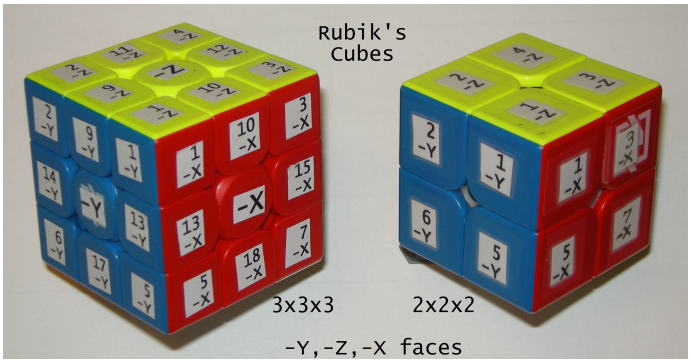
```
01/24/2025  06:12 AM           676,075 MS0-Dir-Doc-txt.pdf           #0
01/23/2025  06:17 PM         1,115,817 MS1-Launch-f95.pdf            #1
01/23/2025  06:18 PM         1,690,381 MS2-RC-NamesEtc-f95.pdf       #2
01/23/2025  06:19 PM         3,655,079 MS3-RC-Workout-f95.pdf        #3
01/23/2025  06:19 PM         1,296,571 MS4-RC-Emulator-f95.pdf       #4
01/23/2025  06:19 PM         1,184,881 MS5-RC-ConCat-f95.pdf         #5
01/23/2025  06:20 PM         2,652,759 MS6-RC-Solve-f95.pdf          #6

***** Be able to easily access this key reference .pdf: *****
01/24/2025  06:12 AM           676,075 MS0-Dir-Doc-txt.pdf           #0
                                  Therein the line of code above is: #0L22
```

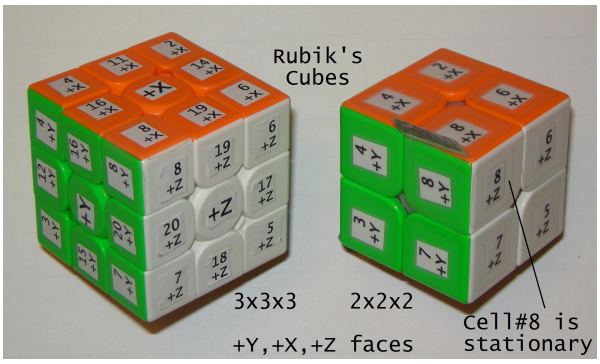
Fortran 95 .pdf's: created using "Absoft Pro Fortran 2022" compiler:

The Fortran source codes also include syntax color-codes. (Red text: comments)

```
01/23/2025  06:17 PM         1,115,817 MS1-Launch-f95.pdf            #1
   Module MS1Def                #1L17+       -basic program variables
   Program MS1Launch            #1L63+       -opens cOutFile
                                             -calls ReadYourNml()
                                             -transfers control to ImportEin()
   Subroutine SaveOutFile       #1L147+      -closes/opens cOutFile
   Subroutine Jdate22()         #1L162+      -World dates & times
   Subroutine nRunSec()         #1L195+      -big, precise interval timers
   Subroutine MyLenTrim()       #1L295+      -truncates strings
   Subroutine Beamer()          #1L280+      -progress bar
                                  ^^^^ : these are .pdf line numbers

01/23/2025  06:18 PM         1,690,381 MS2-RC-NamesEtc-f95.pdf       #2
   Module MS2RCDef              #2L14+       -Min-Steps & RC variables

01/23/2025  06:19 PM         3,655,079 MS3-RC-Workout-f95.pdf        #3
   Subroutine MinStepSolver()   #3L21+       -Min-Steps Lookup tables generator
                                               ...opens & writes many files
   Subroutine BitLog()          #3L548+      -Results Novelty tracker
   Subroutine ReadYourNml()     #3L639+      -called by MS1Launch at the outset.
                                             -opens 'MS4.ini' & EnvNml
   Subroutine ImportEin()       #3L713+      -internalizes .nml data
                                             -branches to apps via %iType
   Subroutine PrintErec()       #3L774+      -view Emulator setup records
                                               the core of Zone(s) setup/use
                                                 and runtine inputs
   Subroutine PrintSRrec()      #3L871+      -view Sequence/Result records
                                               the core of Lookup table gen.

01/23/2025  06:19 PM         1,296,571 MS4-RC-Emulator-f95.pdf       #4
   Subroutine EmulatorRC()      #4L12+       -cell-move-based Emulator
                                             -uses Data- Emulator.txt
   Recursive Function Indexer()#4L116+       -uses Data-Indexer.txt

01/23/2025  06:19 PM         1,184,881 MS5-RC-ConCat-f95.pdf         #5
   Subroutine Concatenate()     #5L14+       -uses Data-nLAV.txt
                                             -uses Data-AttConcat.txt
   Subroutine AllMoves()        #5L106+      -concatenation-based Emulator
   Subroutine ComputeOrder()    #5L260+      -order(s) of Zone subsets

01/23/2025  06:20 PM         2,652,759 MS6-RC-Solve-f95.pdf          #6
   Subroutine GenScramble()     #6L15+       -uses Data-Emulator.txt
   Subroutine SolveScramble()   #6L98+       -exercises Zones, solves 2x2x2's
                                             -uses Data-Emulator.txt
   Subroutine AtoD()            #6L422+      -"Attitudes-to-Discords", i.e.:wisdom
                                             -uses the Discord.bim files
               6 File(s)     11,595,488 bytes
```

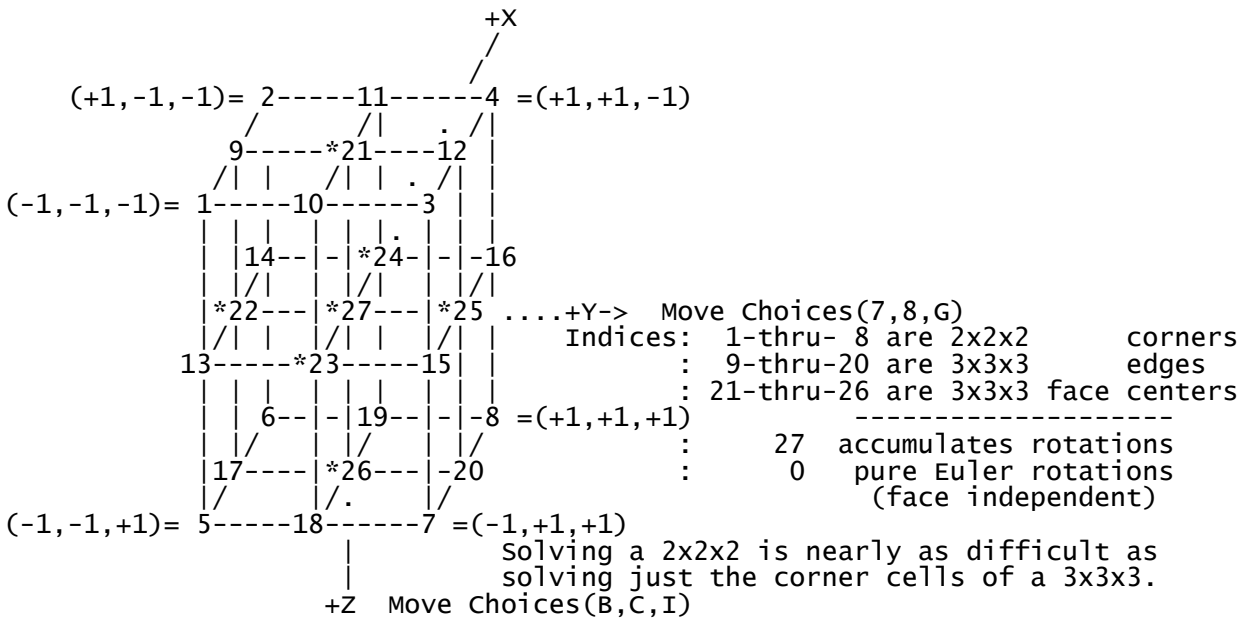
The Voters V are the Rubik's Cube cells.





They are numbered by their "solved" locations:

Cube:	normally "vote":	usually don't vote:	
2x2x2	1: 7	8	no inherent frame of reference. Hence 8 can be fixed.
3X3x3	1:20	21:26	on-axis cells do rotate.
		27	an imaginary cell at the center of the cube.



The variable "nL" [1:27] is used herein to describe the locations above;
don't confuse nL & L, which is the Lookup table variable.

Attitude(s) A:

Link: #0L48 "Data-Attitudes.txt" in .zip

Each cell of a Rubik's Cube has 24 reachable Attitudes designated by the lowercase letters "a" through "x":

#	Letter	Roll	Pitch	Yaw
1	a	0	0	0
2	b	-90	0	0
3	c	+90	0	0
4	d	0	-90	0
5	e	0	+90	0
6	f	0	0	-90
7	g	0	0	+90
8	h	180	0	0
9	i *	0	+180	0
10	j	0	0	180
11	k	-90	0	-90
12	l	-90	0	+90
13	m	0	-90	-90
14	n	0	-90	+90
15	o	0	+90	-90
16	p	0	+90	+90
17	q	+90	0	-90
18	r	+90	0	+90
19	s	-90	0	180
20	t	0	-90	180
21	u	0	+90	180
22	v	+90	0	180
23	w	180	0	-90
24	x	180	0	+90

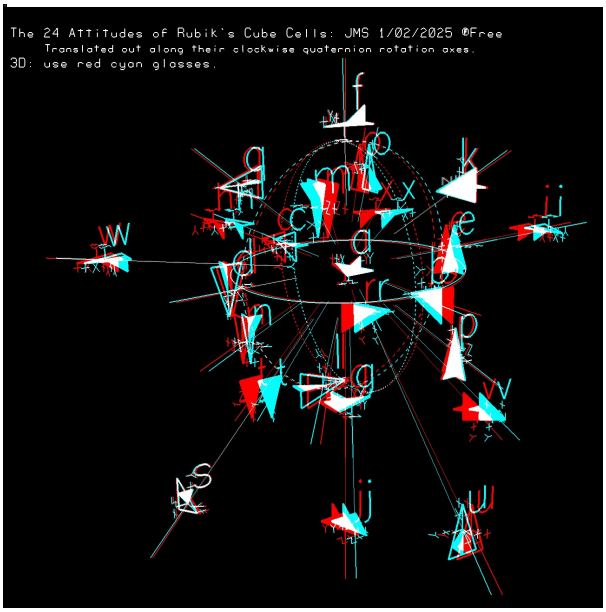


Image: Attitudes-3D.jpg
Use red|cyan glasses

*: Pitch=180.
is non-standard but descriptive,
& i groups with 180 rotations h & j
& follows 1-moves b through g
& precedes 2-moves k through r
& 3-moves s through x.

As a standard aircraft attitude:

		Roll	Pitch	Yaw
9	i *	180	0	180

which would be a Rubik 4-move.

I'll introduce you to a big-data result on the next page: the Discords of all possible 2x2x2 Rubik's Cube [a:x] scrambles:

A Big-Data Result & Challenge #1:

Link: #0L77 "R2C-1234567-6-RrAscii.txt" in .zip

There are 3,674,160 valid scrambles of a 2x2x2 RC, including solved. The link (#0L63) lists them all; the first 10 and last 10 scrambles are shown here. Only a third of the 11,022,480 addresses are populated with valid scrambles & included in the file.

```
R2C-1234567-6-RrAscii.txt
  2  7      1  2  3  4  5  6  7  RubSize, Vtot, V(1:Vtot)
   11022480              3674160 14  Rtot,Stot,Mtot
          1 aaaaaaa                1  0
          4 aaaaajj           3650493 13  1 1 A 1 5 9 5 5 9 6 A 1 6
          8 aaaaakp           2815133 12  1 1 5 1 6 9 2 A 5 A 5 9
         11 aaaaalo           3669095 13  1 5 9 6 1 9 2 A 1 9 5 A 1
         15 aaaaamr           3655474 13  1 5 1 1 5 1 5 5 2 5 9 1 A
         18 aaaaanq           2822174 12  1 1 5 1 A 1 6 A 5 2 5 9
         19 aaaabac           3664953 13  1 5 2 9 9 1 9 5 5 9 5 A 5
         23 aaaabdo            109611  8  5 A 2 9 6 A 1 9
         27 aaaabfr             51338  8  1 5 1 9 2 A 2 6
         29 aaaabkg           3628805 13  1 1 6 2 9 1 9 6 A 2 5 A 6

         ~~ ~~~~~~~           ~~~~~~~ ~~  ~~~~~~~~~~~~~~~~~~~~~~~~~

   11022453 xxxuwqw           3315918 12  1 6 1 6 2 5 2 A 5 5 1 A
   11022456 xxxuwun           1908821 11  1 6 A 5 9 9 5 1 6 9 5
   11022457 xxxuwvh            462646  9  9 6 1 9 6 1 A 6 1
   11022461 xxxuwxl           1783328 11  1 5 A 2 6 9 1 6 A 5 1
   11022463 xxxuxhb            472238  9  9 9 1 6 9 1 5 5 A
   11022467 xxxuxpd            378294  9  6 1 1 5 1 A 6 1 A
   11022471 xxxuxrw            491336  9  A 1 9 9 2 5 5 2 9
   11022474 xxxuxum           3119325 12  1 5 1 5 A 5 A 5 5 9 1 6
   11022475 xxxuxvi           3016434 12  1 1 9 5 9 6 1 6 1 6 2 6
   11022479 xxxuxxk           3001419 12  1 1 9 2 9 5 1 1 A 6 2 9
                  a                    ^^: Discord(s) [0:14]
            ^^^^^^^^: scramble(s) 7[a:x]
            12345678
   ^^^^^^^^: Lookup table address
```

Challenge#1:

The 7-Attitude scramble is the votes of all the Voters & the Discord is the distance from solved.

Find a clear, compact predictive model:

Discord = model(scramble)

…which supercedes this huge Lookup table.

Choice(s) C:

The 18 Move Choices **C** are cube face rotations(degrees):

cube axis: face#:		Choice#'s:							similar to		
		decimal[1:18]:			symbolic[1:I]:				Attitude[b:h]:		
Axis Color		-90	+90	180	-90	+90	180		-90	+90	180
-X (red)	#1	1	2	13	1	2	D		b	c	h
+X (orange)	#2 *	3	4	14	3	4	E		b	c	h
-Y (blue)	#3	5	6	15	5	6	F		d	e	i
+Y (green)	#4 *	7	8	16	7	8	G		d	e	i
-Z (yellow)	#5	9	10	17	9	A	H		f	g	j
+Z (white)	#6 *	11	12	18	B	C	I		f	g	h

^ ~extended but axis

*:omitting #2,#4,#6 hexadecimal specific

leaves Voters/cells Moves listings

8,16.19.20 | & 24.25,& 26

stationary

C#	Symbol	Roll	Pitch	Yaw	Axis:	& as concatenations for cells [1:20]	C# inverses	
0		0	0	0	no move	“aaaaaaaaaaaaaaaaaaaa”	0	=“solved”
1	1	-90	0	0	-X	"babababaabaababaabaa"	2	= ±90°
2	2	+90	0	0	-X	"cacacacaacaacacaacaa"	1	
3	3	-90	0	0	+X	"abababababaababaaba"	4	
4	4	+90	0	0	+X	"acacacacaacaacacaaca“	3	
5	5	0	-90	0	-Y	"ddaaddaadaaaddaadaaa“	6	
6	6	0	+90	0	-Y	"eeaaeeaaeaaaeeaaeaaa“	5	
7	7	0	-90	0	+Y	"aaddaaddaaadaaddaaad“	8	
8	8	0	+90	0	+Y	"aaeeaaeeaaaeaaeeeaae“	7	
9	9	0	0	-90	-Z	"ffffaaaaffffaaaaaaaa“	10	
10	A	0	0	+90	-Z	"ggggaaaaggggaaaaaaaa“	9	
11	B	0	0	-90	+Z	"aaaaffffaaaaaaaaffff“	12	
12	C	0	0	+90	+Z	"aaaaggggaaaaaaaagggg“	11	___
13	D	180	0	0	-X	"hahahahaahaahahaahaa“	13	= 180°
14	E	180	0	0	+X	"ahahahahaahaahahaaha“	14	
15	F	0	+180	0	-Y	"iiaaiiaaiaaaiiaaiaaa“	15	
16	G	0	+180	0	+Y	"aaiiaaiiaaaiaaiiaaai“	16	
17	H	0	0	180	-Z	"jjjjaaaajjjjaaaaaaaa“	17	
18	I	0	0	180	+Z	"aaaajjjjaaaaaaaajjjj“	18	

1 2

12345678901234567890

A concatenation-based Emulator will be introduced on page 13. Thus scrambles can be pre-concatenated and post-concatenated by C’s and C-inverses, so there are a total of 36 move Choices.

2x2x2’s are solved by rotating only the -X,-Y, & -Z faces. Hence there are 9 move choices for both pre- & post-concatenation, and Discords predict the solution paths wisely in that non-intuitive 18-choice environnment.

Results address a_ Indexer:

The code is in #4L116-367 – Recursive Function Indexer(): computes
Lookup table addresses recursively .

"Recursion" is when a subroutine or function calls itself. In the example below the addressing is 20-dimensional, & 'next-address' code recurs 20 times; the address accumulates during de-recursion just before each recursion returns to the recursion that called it. The full address pops out of the top recursion as a function result!

The Emulator, processing the Choices & computing Voters' Attitudes determine how many Results there will be. Hence addressing Results is problem-dependent.

Here's the worst-case scenario: For the 3x3x3 RC there are 20 cells(Voters) with nominally 24 attitudes each. If all the attitudes were reachable by all the cells then: the total addresses = 24^20 = 4.019988e27 ~4.01 septillion.

Fortunately, each located corner cell removes three attitudes from consideration, and each located edge cell removes two attitudes from consideration. The Indexer recursively computes the needed address space size to be: 5.19e20 as follows:

```
full-in  : 20 20  1                                                    1 edges
de-recur : 20 20  2 x                   2   * 0 + ^> = ~ single choice   1
de-recur : 19 19  4                     8   * 0 + ^> =                   1
de-recur : 18 18  6                    48   * 0 + ^> =                   1
de-recur : 17 17  8                   384   * 0 + ^> =                   1
de-recur : 16 16 10                  3840   * 0 + ^> =                   1
de-recur : 15 15 12                 46080   * 0 + ^> =                   1
de-recur : 14 14 14                645120   * 0 + ^> =                   1
de-recur : 13 13 16              10321920   * 0 + ^> =                   1
de-recur : 12 12 18             185794560   * 0 + ^> =                   1
de-recur : 11 11 20            3715891200   * 0 + ^> =                   1
de-recur : 10 10 22           81749606400   * 0 + ^> =                   1
de-recur :  9  9 24         1961990553600   * 0 + ^> =                   1 corners
de-recur :  8  8  3 x       5885971660800   * 0 + ^> =   single choice   1
de-recur :  7  7  6        35315829964800   * 0 + ^> =                   1
de-recur :  6  6  9       317842469683200   * 0 + ^> =                   1
de-recur :  5  5 12      3814109636198400   * 0 + ^> =                   1
de-recur :  4  4 15     57211644542976000   * 0 + ^> =                   1
de-recur :  3  3 18   1029809601773568000   * 0 + ^> =                   1
de-recur :  2  2 21  21626001637244928000.0 * 0 + ^> =  *                 1.0
de-recur :  1  1 24 519024039293878272000.0 * 0 + ^> =  *                 1.0
                  \sx\qn\qd\tr\bl\ml\th\hn\         \sx\qn\qd\tr\bl\ml\th\hn\
                  ^24^21^18^15^12^9 ^6 ^3 ^0        ^24^21^18^15^12^9 ^6 ^3 ^0

                  ^^:effective number of Choices         *: 64-bit integers fail,
               ^^:cell#                        used the mantissas of 128-bit reals.
```

4.019988e27 / 5.19024e20 -> 7,745,283 times smaller address space is needed, and probably only 1/6th of these are valid scrambles. However, needing only a 5.19 quintillion byte novelty checker drove me to consider a more compact 3x3x3 3-Zone solution. Zones #2 & #3 don't cooperate yet, and the Zone#1 quad solves oblivious to how bad the scramble will be for the other two Zones.

Zone(s) **Z**:

Within a Zone the Lookup table of Discord values is useless without the addresser. Several components must interact. Given Voters with Attitudes **Rp**, the Emulator processes Choices creating **R'** . Each Zone has its own parameters for using the Indexer to compute **aR'** & hence Lookup the Discord **Z**(**D**)= **Z**(**L**(**aR'**)). The Indexer and Emulator are dual-use, in that they are used to generate the Discord Lookup table for a single Zone and subsequently exercised to report multi-zone Discords. The 2x2x2 RC is solved by a single zone - #2. Subroutine AtoD() can retrieve the Discords of up to nine Zones for a given scramble.

The three zones and run summaries are defined in MS-RC.zip by:

Rubik forward-right-down cells quadrant cells[8,16,19,20]:

#0L69 R3Q-08161920-12.nml <-sets up the Ein% record
#0L73 R3Q-08161920-12-Summary.txt <-run summary text file
& Ein% Zone printout

Rubik non-quad corners cells[1,2,3,4,5,6,7]: [8] is fixed.

#0L75 R2C-1234567-6.nml This is the only Zone for 2x2x2 solving.
#0L77 R2C-1234567-6-RrAscii.txt <-the challenge-file on page 8
#0L79 R2C-1234567-6-Summary.txt

Wherein Min-Steps novelty testing reveals the follow 14-level Discord profile:

nSeq,	n1st,	nLast,	count	Level	<sec>	Run
0	1	1	1	0.000		0.000
1	2	7	6	0.015		0.015
2	8	34	27	0.000		0.015
3	35	154	120	0.016		0.031
4	155	688	534	0.047		0.078
5	689	2944	2256	0.141		0.219
6	2945	11913	8969	0.515		0.734
7	11914	44971	33058	1.891		2.625
8	44972	159120	114149	6.595		9.220
9	159121	519628	360508	21.453		30.673
10	519629	1450216	930588	65.354		96.027
11	1450217	2801068	1350852	156.052		252.079
12	2801069	3583604	782536	212.598		464.677
13	3583605	3673884	90280	118.381		583.058
14	3673885	**3674160** ^	276	13.580		596.638
15		276	6648.012/sec	est.=		0.042
if real:		**11022480**	6158.106/sec	est.=		1789.914

^: 2/3rd's are not mechanically reachable.

Rubik non-quad edges cells[9,10,11,12,13,14,15,17,18]:

#0L81 R3E-091011121314151718-6.nml
#0L85 R3E-091011121314151718-6-Summary.txt

The summary file(s) also show the first pass of recursions of the address Indexer for each Zone.

Emulator E:

Move Choice Emulator:

This Emulator exploits the fact that cell Attitude also specifies cell location, and Choices act on cells currently occupying a specific face. Hence a cell/Voter number and current Attitude combine with a current Choice to predict the resulting Attitude output which also infers the revised location output. A big three-argument array stores the details:

```
AECAV(C,A,V)
          ^:cell/Voter# [1:27]
        ^:Attitude      [1:24]  =[a:x]
      ^:Choice          [0:19]  =[1:I+1]  =19 is an error handler
  ^:Emulator
^:Attitude-out          [1:24]  =[a:x]
```

"AECAV" = Attitude <- Emulator <- Choice + Attitude Voter
out the array iin in in

Here are the array's values for Cell#1 (24 lines of numbers) :

```
    |     |Choice        [D:I] 180 deg rotations:-X,+X,-Y,+Y,-Z,+Z
 nV|nAin |0  1  2  3  4  5  6  7  8  9  A  B  C| D  E  F  G  H  I er
  1   1   1  2  3  1  1  4  5  1  1  6  7  1  1  8  1  9  1 10  1  1
  1   2   2  8  1  2  2 13 16  2  2  2  2 11 12  3  2 22  2  2 19  2
  1   3   3  1  8  3  3  3  3 14 15 17 18  3  3  2  3  3 19 22  3  3
  1   4   4  4  4 12 17  9  1  4  4 13 14  4  4  4 21  5  4 20  4  4
  1   5   5 11 18  5  5  1  9  5  5  5  5 15 16 20  5  4  5  5 21  5
  1   6   6 13 15  6  6  6  6 17 11 10  1  6  6 24  6  6 23  7  6  6
  1   7   7  7  7 16 14 12 18  7  7  1 10  7  7  7 23 24  7  6  7  7
  1   8   8  3  2  8  8  8  8 20 21  8  8 23 24  1  8  8 10  8  9  8
  1   9   9  9  9 19 22  5  4  9  9  9  9 24 23  9 10  1  9  9  8  9
  1  10  10 10 10 22 19 10 10 21 20  7  6 10 10 10  9 10  8  1 10 10
  1  11  11 20  5 11 11 11 11  6 23 11 11 19  2 18 11 11 17 11 12 11
  1  12  12 12 12 21  4 24  7 12 12 12 12  2 19 12 17 18 12 12 11 12
  1  13  13 24  6 13 13 22  2 13 13 20  4 13 13 15 13 16 13 14 13 13
  1  14  14 14 14  7 23 14 14 19  3  4 20 14 14 14 16 14 15 13 14 14
  1  15  15  6 24 15 15 15 15  3 19 15 15 21  5 13 15 15 14 15 16 15
  1  16  16 16 16 23  7  2 22 16 16 16 16  5 21 16 14 13 16 16 15 16
  1  17  17 17 17  4 21 17 17 23  6 22  3 17 17 17 12 17 11 18 17 17
  1  18  18  5 20 18 18  7 24 18 18  3 22 18 18 11 18 12 18 17 18 18
  1  19  19 19 19 10  9 19 19 15 14 19 19 12 11 19 22 19  3 19  2 19
  1  20  20 18 11 20 20 20 20 10  8 14 13 20 20  5 20 20 21  4 20 20
  1  21  21 21 21 17 12 21 21  8 10 21 21 16 15 21  4 21 20 21  5 21
  1  22  22 22 22  9 10 16 13 22 22 18 17 22 22 22 19  2 22  3 22 22
  1  23  23 23 23 14 16 23 23 11 17 23 23  9  8 23  7 23  6 23 24 23
  1  24  24 15 13 24 24 18 12 24 24 24 24  8  9  6 24  7 24 24 23 24
  2   1   1  1  1  2  3  4  5  1  1  6  7  1  1  1  8  9  1 10  1  1
  2   2   2  2  2  8  1 13 16  2  2  2  2 11 12  2  3 22  2  2 19  2
  2   3   3  3  3  1  8  3  3 14 15 17 18  3  3  3  2  3 19 22  3  3
  2   4   4  4  4 12 17  9  1  4  4  4  4 13 14  4 21  5  4  4 20  4
```

Each of the other 26 cells has a corresponding coefficient block.

This approach supports tracking arbitrary subsets of cells.

Link: #0L44 "Data-Emulator.txt" in .zip
& : #4L12-115 "Subroutine EmulatorRC()"

Concatenation Emulator:

This Emulator exploits the fact that scrambles can be connected to one another in a continuous way. The left scramble is reordered to match the output locations (nLAV below) of the right scramble after which the right scramble's attitudes are modified by the left scrambles attitudes. Two compact arrays and seven lines of code accomplish this:

The first array maps Voter Attitudes to Locations. nLAV(1:24,0:27)
Link: #0L46 "Data-nLAV.txt" in .zip nL = nLAV(A , V)

The second array concatenates any two attitudes. AoutAAprev(1:24, 1:24)
Link: #0L43 "Data-AttConcat.txt" in .zip Aout = AoutAAprev(A ,Aprev)

The code is in #5L76-83 of Subroutine Concatenate():

```
76  !Concatenation is accomplished in the next 7 lines of code:
77   do nV = 1,VtotL
78     nVAprev(     nV) = nLAV(VAprev(nV),nV)
79     VAnextmod(   nV) = VAnext(nVAprev(nV))
80     if(VAprev(   nV) == 0) cycle
81     if(VAnextmod(nV) == 0) cycle
82     VAnewL(nV) = AoutAAprev(VAnextmod(nV),VAprev(nV))
83     enddo!nV
```

```
An example:
Concatenate in previous`s order: ----------
      nV : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Move Choice *2's scramble:
A VAprev : 3 1 3 1 3 1 3 1 1 3 1 1 3 1 3 1 1 3 1 1 :previous
         : c a c a c a c a a c a a c a c a a c a a
Move Choice #6's scramble:
A VAnext : 5 5 1 1 5 5 1 1 5 1 1 1 5 5 1 1 5 1 1 1 :pre-morph
         : e e a a e e a a e a a a e e a a e a a a

n nVAprev : 3 2 7 4 1 6 5 8 9 15 11 12 10 14 18 16 17 13 19 20 :morph

A VAnextmod: 1 5 1 1 5 5 5 1 5 1 1 1 1 5 1 1 5 5 1 1 :post-morphed
           : a e a a e e e a e a a a a e a a e e a a
Resulting scramble:
A VAnew : 3 5 3 1 15 5 15 1 5 3 1 1 3 5 3 1 5 15 1 1 :concatenation
        : c e c a  o e  o a e c a a c e c a e  o a a
                    ____________________
```

Array AECAV(C,A,V), the Move Choice Emulator on the previous page, produces the same result, but can only left-concatenate single moves:

```
Choice: Scramble:
   0  : aaaaaaaaaaaaaaaaaaaa
   2  : cacacacaacaacacaacaa
   6  : cecaoeoaecaacecaeoaa
```

R2 Zone use- The 2x2x2 Solver:

After unpacking #0L10 MS-RC-Zip & renaming MS-RC-64.exf to MS-RC-64.exe:
The previous Zone use run summary is:
#0L62 R2-ZoneUse-Summary.txt -will be overwritten if UtIn=13.

Set: #0L61 R2-ZoneUse.nml
as the first line of: #0L53 MS3.ini
and then run: #0L40 MS-RC-64.exe

```
The first run prompt is:
    Use this free software at your own risk:
       Press enter to continue           -or-
       Close this command prompt window.

After the Zone for 2x2x2 corners is loaded, the instructions are:
||||||||||||||||||||||||||||||||||||||||||||||||||
Move = 0 closes this program.

Negatives of the Ein%MCvis Move Sequence values above will deconstruct
the scramble to solved, demonstrating post-multiplication.

Otherwise, pick any move that reduces the Discord of a Zone, &
thus the 2x2x2 RC will solve in a mimimum number of moves.

Move#0 shows the individual zone Discord(s) and the order(s).
          Order: the number of self-concatenations to solved.

now = ____ sums the values of Zone#2 & #3 for 3x3x3`s.
||||||||||||||||||||||||||||||||||||||||||||||||||
Previous move [-18:18] =  0
cVAin =             lgpebela.............  <- the present scramble
Ztot = 1             Discord:  Order:
Move#  Attitudes:    Zone#:    1 :    1
-17 -H tmfqbela.............   12      7
-15 -F dspewkla.............   12      9
-13 -D rgceoera.............   12     15
-10 -A bdkabela.............   13     18
 -9 -9 jruvbela.............   13     36
 -6 -6 napeslla.............   11      6
 -5 -5 kbpeipla.............   11     30
 -2 -2 hgxexeua.............   13     36
 -1 -1 eggegeaa.............   11      6

  0  0 lgpebela.............   12      7   <- Move = 0 closes the program

  1 +1 lgwkheua.............   11     36
  2 +2 lggraeda.............   11      6
  5 +5 xgbebaxa.............   11      6
  6 +6 ggvebiga.............   11     30
  9 +9 lapoboba.............   11     18
 10 +A ljppbpsa.............   11     36
 13 +D lgntceqa.............   12     15
 15 +F rgmebdra.............   12      9
 17 +H lfpubuka.............   12      7
                        max:  14           Any one of 9 moves will reduce the discord to 11:
        Pre-multiplies:                Post-multiplies:          -6.-5,-1,+1,+2,+5,+6,+9,+10
Move#`s:  1   2  13D     3   4  14E     -1  -2 -13     -3  -4 -14
          5   6  15F     7   8  16G     -5  -6 -14     -7  -8 -16
          9  10A 17H    11B 12C 18I     -9 -10 -17    -11 -12 -18
   -:   -90 +90 180 +: -90 +90 180 -:  -90 +90 180 +: -90 +90 180
   X:    11  11  12      0   0   0      11  13  12      0   0   0 :Z#1
   Y:    11  11  12      0   0   0      11  11  12      0   0   0  now =   12
   Z:    11  11  12      0   0   0      13  13  12      0   0   0
||||||||||||||||||||||||||||||||||||||||||||||||||
Next move [-18:18] =
```

R3 Zone use – 3x3x3's: after unpacking the .zip file #0L10

The previous Zone use run summary is:

	#0L66 R3-ZoneUse-Summary.txt	<- two excerpts are shown below:
Set:	#0L65 R2-ZoneUse.nml	known post-multiplies solving Zone#1:
as the first line of:	#0L53 MS3.ini	
and then run:	#0L40 MS-RC-64.exe	

```
||||||||||||||||||||||||||||||||||||||||||||||||||
Previous move [-18:18] =  0
cVAin =            gjwmwovndrjnrbkvsvhv          a Zone#1, #2,& #3 scramble:
The current scramble, pre-mult`s on left, post-mults on right:
Moves:(23:1)  =    +D,+A,+5,+2
                  ,+A,+D,+5,+1,+A,+2,+5,+A,+1,+5,+1,+D,+6,+1,+G,+9,+1,+7,+3,
||||||||||||||||||||||||||||||||||||||||||||||||||
Previous move [-18:18] =-16
cVAin =            lshlrvnaxlolxvwasxaa           reduced to a Zone#2 & #3 scramble:
The current scramble, pre-mult`s on left, post-mults on right:
Moves:(28:1)  =    +D,+A,+5,+2,+A,+D,+5,+1,+A
                  ,+2,+5,+A,+1,+5,+1,+D,+6,+1,+G,+9,+1,+7,+3,-3,-7,-1,-9,-G,
Ztot = 3                      Discord:          Order:
Move#  Attitudes:     Zone#:   1   2   3    :    1    2    3
-18 -I lshljobkxlolxvwajjwc    8  -1  -1         60   30   12
-17 -H qicqrvnaqnqwxvwasxaa    0  11  15          1   30   12
-16 -G lsiprvkjxloixvigsxak    6  -1  -1         24    8   12
-15 -F cqhlbknablolcfwafxaa    0  11  15          1   12    6
-14 -E lhhsrrnvxlhlxhwssxpa    8  -1  -1         24    8    6
-13 -D msllavraxgolfvgasraa    0  11  15          1   12    8
-12 -C lshlcefpxlolxvwaifnf    7  -1  -1        210   21   20
-11 -B lshlmgtsxlolxvwagogh    8  -1  -1         30    6   20
-10 -A xddnrvnaddisxvwasxaa    0  12  14          1    7   10
 -9 -9 ouuwrvnachuuxvwasxaa    0  12  16          1    6   10
 -8 -8 lssdrvuwxlodxvsdsxap    6  -1  -1         10   15   20
 -7 -7 lsgtrvebxlonxvebsxae    5  -1  -1         30   45   20
 -6 -6 krhlscnarlolokwaoxaa    0  12  16          1   12   12
 -5 -5 vbhlqlnamlollmwaqxaa    0  12  14          1   12   12
 -4 -4 ljhirbnxxljlxbwesxba    8  -1  -1         60   45   12
 -3 -3 lghcrjnixlclxuwcsxia    4  -1  -1         60   15   12
 -2 -2 gsxldvcaxrolrvxasqaa    0  10  16          1   12   12
 -1 -1 bstlgvxaxkolgvlaslaa    0  12  16          1   12   12

  0  0 lshlrvnaxlolxvwasxaa    0  11  15          1   18    4

  1 +1 ljclrigaololoinasxaa    0  12  14          1   12   12
  2 +2 liblrjwamlolmjpasxaa    0  10  14          1   12   12
  3 +3 ushuevnbxuouxvwbsxba    1  -1  -1         60   45   12
  4 +4 dshdtvncxdodxvwcsxca    1  -1  -1         60   15   12
  5 +5 xstlgvsaxxolrvwaoraa    0  12  14          1   12   12
  6 +6 gsulxvcaxgollvwanlaa    0  12  14          1   12   12
  7 +7 lohxrpndxlclxvkdsxad    1  -1  -1         10   15   20
  8 +8 lnhgrmnexlslxvqesxae    1  -1  -1         30   45   20
  9 +9 lshbcrdaxlubxrwashaa    0  12  16          1    7   10
 10 +A lshsvqtaxlesxqwasiaa    0  12  16          1    6   10
 11 +B blwlrvnfhlolxvwalxff    1  -1  -1        210   21   20
 12 +C skxlrvngilolxvwakxgg    1  -1  -1         30    6   20
 13 +D lvalrspaflolfsgasxaa    0  11  15          1   12    8
 14 +E qshqkvnhxqoqxvwhsxha    2  -1  -1         24    8    6
 15 +F rsjllvoaxrolgvwacgaa    0  11  13          1   12    6
 16 +G lchrrbnixlnlxvfisxai    2  -1  -1         24    8   12
 17 +H lshkqcmaxlpkxcwaswaa    0  13  15          1   30   12
 18 +I kbilrvnjwlolxvwabxjj    2  -1  -1         60   30   12
                         max:  9  14  18
       Pre-multiplies:                Post-multiplies:
Move#`s:  1   2  13D     3   4  14E     -1  -2 -13     -3  -4 -14
          5   6  15F     7   8  16G     -5  -6 -14     -7  -8 -16
          9  10A 17H    11B 12C 18I     -9 -10 -17    -11 -12 -18
   -:   -90 +90 180 +: -90 +90 180 -:  -90 +90 180 +: -90 +90 180
   X:     0   0   0      1   1   2       0   0   0      4   8   8 :Z#1
   Y:     0   0   0      1   1   2       0   0   0      5   6   6  now =    0   solved
   Z:     0   0   0      1   1   2       0   0   0      8   7   8
          ----------     ----------     ----------     ----------
   X:    26  24  26     -1  -1  -1      28  26  26     -1  -1  -1 :Z#2+Z#3
   Y:    26  26  24     -1  -1  -1      26  28  26     -1  -1  -1  now =   26
   Z:    28  28  28     -1  -1  -1      28  26  26     -1  -1  -1
||||||||||||||||||||||||||||||||||||||||||||||||||
Previous move [-18:18] =  0
```

This solves the remaining edges, or remaining corners, but not both!
… which introduces a niftier 2nd challenge:

Challenge #2: Realizing uncommon "common sense":

"Common sense" is poorly defined, but involves being aware enough to avoid catastrophes caused by the unforeseen secondary and tertiary consequences of personal decisions. Making CAD-assisted choices which avoid catastrophes implicitly will eclipse common sense. To do that, multiple wisdom Zones need to be navigated simultaneously.

'Zone use - 3x3x3' on the previous page realizes a decision environment in which there are three Zones of wisdom. Zones #1, #2, & #3 are separately wise within their Zones. However, in my experiments, they have not yet solved deep scrambles (e.g. 40 random 18-Choice Moves), let alone solve them in a minimum number of steps.

For the 3x3x3 Rubik`s Cube: three wisdom Zones now exist (see page 11), so challenge #2 is a defined problem:

Challenge #2:

Demonstrate unifying multiple Zones into a minimum-step 3x3x3 Rubik's Cube solution within a WisdomCAD environment.

Perhaps the solution algorithms are already known,
e.g.: by people who program auto navigators.

The future offers CAD-based wisdom, even for hobbyists.
Efficient earthly cooperation is imaginable.

Afterthoughts:

1. 'Min-Steps', this paper, is a 'discrete optimal control' paradigm.

2. "WisdomCAD" =
"The unification of 'the common good', optimal control, and CAD."

3. Challenge #3: Demonstrate Artificial Intelligence ('AI')
contributing to World Peace
in a learnable way.

4. Challenge #4: "Begin working on WisdomAI"
Pry 'intelligence' from the grip of guile using artificial means.

5. Given what I learned: "I was dumber than a Rubik's Cube" lingers as an amusing refrain. My thanks to Mr. Rubik, for his fully-cooperative puzzle.