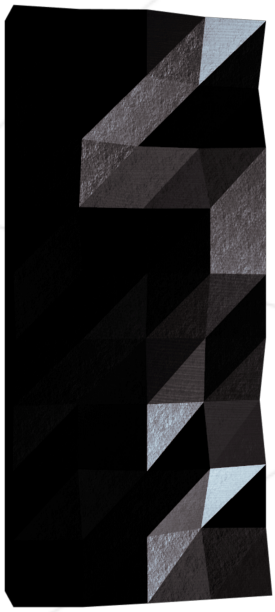

Rubik

Anything planned for next wednesday?

42 Staff pedago@staff.42.fr

Summary: This project will make you write a program that solves Rubik's Cubes with minimum spins.

Version: 4.1

Contents

Chapter I

Preamble

The following are some realistic examples that you should improve:

https://www.youtube.com/watch?v=eQH7MU0gUCQ

It looks simple:

https://www.youtube.com/watch?v=_v85dcvw2vQ

But this is way too easy:

https://www.youtube.com/watch?v=X0pFZG7j5cE

Chapter II

Introduction

In this project, you're gonna have to solve Rubik's Cubes under official Rubik's Cubes competition conditions, and the Fewest Moves Challenge (FMC) in particular, minus the usual hour limit (you will have in fact some seconds never beyond 3 seconds).

During a Fewest Moves Challenge, competitors are locked in a room with a piece of paper, a pencil and a cube. That's it. They will have one hour to find the shortest solution to their cubes and lay it on paper. Solutions will be individually verified afterwards, and the lowest score will be the winner.

Good luck!

The current FMC record is 20 spins. It is held by Tomoaki Okayama. More than 200 persons have already found solutions below 30 spins in competition.
Moreover, it's been proven that any cube can be solved in 20 spins max. More infos: http://www.cube20.org/

It is strongly advised that you learn the basics of Rubik's Cubes IRL before your take this project.

Chapter III

Objectives

Everybody knows the Rubik's Cube, and chances are you already lost bits of your brains trying to solve one.

Whether your managed to solve it, this project can be tackled by any motivated programmer. Indeed, it's just a little algorithmic project that will require a light skill in space visualization, moderate notions of group theory. Grey cells is a plus. This is odds and ends to you.

Chapter IV

General instructions

You're free to use the language you like (C, C++, Perl,Python, Java, Brainfuck if you think it's funny?). No segfault, memory leak, double free, infinite loop. There will be no tolerance for unreasoned interruption, mainly because this should clearly not be an issue with this project.

If appropriate, you will have to provide a Makefile with the usual rules to compile your program.

You can use any library as long as you can justify them during the evaluation. Of course, a library or an external resource that does the work for you is not authorized. You will have to justify your own alogrithm during the evaluation.

You must be able to explain your algorithm with simple words and visual concepts.

Chapter V

Mandatory part

Your program must accept a mix in parameter.

A mix is a spin sequence separated by one or several spaces applied on a cube as to mix it, not randomly, but in a predetermined pattern.

The notation used is the one used globally (F R U B L D for Front / Right / Up / Back / Left / Down). To understand this notation, you can go to http://www.francocube.com/notation.php (FIND AN ENGLISH WEBSITE). This is a good starting point, the rest of the website providing a lot of informations for this project. You should also watch the project video very carefully.

You must not use any alternative notation system. It would be confusing during evaluation and would bore everyone.

F R U B L D represents both a face and a spin applied to this face, but the difference usually is rather explicit. Your programs must accept spin modifiers and send a solution that uses this notation.

Here is an example of a valid sequence: R2 D' B' D F2 R F2 R2 U L' F2 U' B' L2 R D B' R' B2 L2 F2 L2 R2 U2 D2

Spins of slices M, E and S as well as rotations of x, y and z are prohibited.

Your program must return the spin sequences on a 3x3x3 cube mixed with the given sequence beforehand on the standard output.

For instance:

```
$>./rubik "F R U2 B' L' D'" | cat -e
```

Returning the opposite of the mix (or direct variant with sequence insertion without any effect to modify the length) will be considered cheating...

As usual, you will have to properly manage errors.

Metrics used to count the spins is a half-turn metrics (HTM), which means a quarter-turn or a half-turn on a same face will count as one spin, but slice spins count for 2. In short, your score will be the result of a wc -w on your solution.

```
$> ./rubik "F R U2 B' L' D'" | wc -w
```

In order to validate this project, it will be necessary to have an average of 50 spins with a maximum response time of 3 seconds.

Chapter VI

Bonus part

Here are a few ideas of bonuses:

- A real graphic represents the cube's position sporadically (ncurses, minilibx, openGL? Your choose, but a series of figures or letters is considered a debug, not a real bonus...). You can also try to graphically show the cube spin in real time (this would be real nice!).
- An algorithm that goes lower than the most optimized solutions in a reasonable time (beyond a few seconds is not considered reasonable).
- A choice between several algorithms, or a selection of the best solution between several algorithms.
- An integrated mixing generator you can specify the length and/or the number.
- A subdivision of your solution in "humanly understandable" substeps.
- Ways to work with other puzzles (4x4x4, 2x2x2, Megaminx, Square-1?)
- Real performances of the evaluated student to the resolution of a Rubik's Cube during the evaluation.

The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.

These bonuses must not go beyond the regular program's operation. (i.e. mix input must always be a valid sequence). Prior options can be indicated and preceded with a '-'.

Chapter VII

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.