Rubik's cube

Interim report on project work for course in Mathematics with Computer Science

Matevž Nolimal

4/2/2022

Project description

The goal of the project is not just to write a program to play, but also solve the Rubik's cube in an efficient way. While writing the code I will restrict myself to follow the so called clean code principles and write clean code that follows most recent Computer Science endeavours. i decided to write the code in Python as it is the close of the programming languages to me and I have gained plenty experience working with it.

I shall do my best to follow the Test Driven Development (in Computer Science more known under the abbreviation TDD). The term TDD stands for coding while restricting the fact that not a single line of production code should be written without the failing test first. Like that the programer make sure it has a fully tested code that matches expection/wanted behaviour. I learned about the clean coding principles from the following book: [1] Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin. Code in Python and will be using the robust code structures and abstructions that have multiple advantages over the old/outdated way of coding.

The goal of Rubik's cube is well known by everyone, but now everyone knows the tricks and obstacles that one encounters while solving the cube by hand.

Work done

Due to the fact that I never truly tacked the problem prior this course, I first had to find/study the material on Rubik's cube solvers. Following many online forums I stumbled upon the Beginner Rubiks Soltuion reference paper [2] which gave me a good understanding of the problem, and a possible the way to solve the cube. What I found great about the paper is that it was written in the "programming way" which enabled me planning the code implementation of the game.

Following the paper's proposal of the terminology I envisioned writing several Python classes, whereas each one defines one layer of information. Writing the code bottom up this implied coming up with a structure/class defining the smallest piece in a cube i.e "Piece". The 3 x 3 x 3 cube consists of 27 such pieces. The most intuitive implementation of such classes that define objects with data and very minimal functionalities in Python are the so called dataclasses. They are constructed by adding the dataclass decorator to the somewhat improved normal class structure. Having done plenty of good experiences with such class structures I often emphasize this is the best way one can build the programs using Python.

Next level that will is required will be "Point" which is a 3d vector defining the position of the before mentioned pieces in the cube. Following the above mentioned paper I quickly realised the main funtionality of the program will be backend that serves/enables rotation of "Points" in different ways along the three axis (x - poiting to the right, y - direction up, z - direction to the front) of the Rubik's cube. This is the part of the code I currently work on. I did some researching on rotation matrices in order to understand what has to be done. I followed the reference [3] which gave me sufficient information on the subject needed for this task.

Next steps

In order to implement the game at last I will still have to combine/implement the above mentioned basic dataclasses to some “Cube” context that will enable the game to be played. Last but not least I will have to implement the solver. It will replicate the layer based solving method, which is introduced with some mock-up examples in the mentioned reference paper [2].

(optional) If time will allow I might as well try to draft if not also implement some of the improvements/shortcuts to minimise the solving time of the chosen solving algorithm.

References

[1] Robert C. Martin: Clean Code: A Handbook of Agile Software Craftsmanship (2008) p. 122-133

[2] Beginner Solution to the Rubik’s Cube (2005) p. 1-7

[3] Rotation matrix https://en.wikipedia.org/wiki/Rotation_matrix