Twisty Puzzle 2.0
(Project Notes)

05/06/2023

- [ ] Clean up the android code
- [ ] Update the project to API 33 and AndroidX
- [ ] Restructure the project .java files to make a CubeSolver module
- [ ] Use proper code style guide for java (reference software engineering document)
- [ ] Make Rayfuzu Learning YouTube video
- [ ] Create a Twisty Puzzle Masterz 2.0 repo on github
- [ ] Create design diagrams and descriptions on exactly how the data structure and algorithms work.

~~**Option 1:**~~
- ~~Android app cube solver working~~
- ~~Port that over to python~~
- ~~Use working OpenCV python code with my working solver~~

Option 2:
- Android app cube solver working
- Add Android Java OpenCV

Video Format Inspiration:

https://www.youtube.com/watch?v=fAX27_FyU9g

Working Java Android Cube Solver and OpenCV
https://github.com/ucchiee/AndroidRubikCubeSolver
Add Open CV to android:
- https://www.youtube.com/watch?v=oIk2hTPxFqs&t=255s
    - At 6:28 whatever you name that module, open that folder in file explorer and paste everything from C:\Users\downs\Desktop\OpenCV-android-sdk**\sdk into it**
    - **Update build.gradle for (:opencv)** add this line

```
android {
    namespace 'org.opencv'
```

    - **Update build.gradle for (:opencv)** with 4 fields to match your project build.gradle
        - a) compileSdkVersion
        - b) buildToolsVersion
        - c) minSdkVersion and
        - d) targetSdkVersion.
    - Comment out each line that is throwing an error inside of **AsyncServiceHelper.java** (I don't use that anywhere)

- https://www.youtube.com/watch?v=bR7IL886-uc&t=221s

- Also helpful
    - https://stackoverflow.com/questions/63254458/could-not-import-the-opencv-library-in-android-studio (last post in this stackoverflow thread)
    - https://www.geeksforgeeks.org/how-to-add-opencv-library-into-android-application-using-android-studio/
    - https://www.geeksforgeeks.org/different-ways-to-delete-a-module-in-android-studio/

Working python OpenCV (using open source solver)
https://github.com/nicpatel963/CubeSolvingScript/blob/master/cubenew.py

Working Android OpenCV and Solver (But using USB cameras?)
https://github.com/geoffreywwang/CubeBot

1. https://www.youtube.com/watch?v=afAGtExoiLQ
2. https://www.youtube.com/watch?v=RMo_CLi1Z5g
3. https://www.youtube.com/watch?v=CWmKHcx1X6A
4. https://www.youtube.com/watch?v=3pqo6SMmtS4

   9 years old. Can't build into Android Project without Gradle.
5. https://sgelb.github.io/projects/arcs

   3D cube animation
6. https://github.com/cjurjiu/AnimCubeAndroid

How to solve the 3x3x3 Rubik's Cube

1. Terminology

a. Location:
   i. All interesting layers for the given cubie

b. Orientation
   i. Placement of the stickers on the given cubie with respect to their correct center piece colors.

2. Cross

a. Solve Order

i. {
   [WHITE, GREEN] , [WHITE, BLUE] , [WHITE, ORANGE] , [WHITE, RED]
}

b. Steps

i. Get all cross pieces on down layer
ii. Rotate bottom layer until at least 2 pieces are in correct location
iii. Set the cubies that are in correct locations to their correct orientation
iv. Iff 2 cubies are in incorrect locations then swap them
   1. Set those 2 cubies to correct orientation Iff need to.

c. Step 1

i. Get all cross pieces on down layer

d. Avoidance Maneuver

This is done by rotating the down layer such that there is no white cubie on the spot that itersets with the down and chosen layer to move.

// Chosen layer to move
n = [L or R]

- D (until no white edge cubie interests [n,D])
- n or n’

    - ***IMPROVEMENT 1:*** *At this point you already know which white edges have been solved so you don’t need to (do while) rotate the down layer until you are above an unsolved piece.*
    - *Instead, you can just keep track of those 4 pieces and which ones are solved and unsolved. That way you can know exactly what movements are needed to put the unsolved white edge piece under your solving cubie.*

    - ***IMPROVEMENT 2:*** *If you have to do D D D then undo those steps and do a D’*

e. If cubie on up layer

[top, right] => R2
[top, front] => F2
[top, left]   => L2
[top, back] => B2

If there is a white edge cubie already located at [n, D] then use the **avoidance maneuver** to solve the cubie in the up layer.

f. **If cubie on middle layer**

[left,	front]	=> L or F’
[right,	front]	=> R’ or F
[left,	back]	=> L’ or B
[right,	back]	=> R or B’

Prefer the move that puts the cubie in the correct <u>location</u>.

(or)

Move that wont kick out an already positioned white cubie on the down layer.

g. **If cubie on down layer**

i. Go to the next white edge cubie to solve. This one is already on the desired layer.
ii. Maintain location on down layer while putting other cubies on down layer using the **<u>avoidance maneuver</u>**.

h. Step 2

i. Rotate bottom layer until at least 2 pieces are in correct location

i. Step 3

i. Set all cubies to their correct orientations.

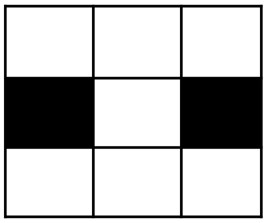
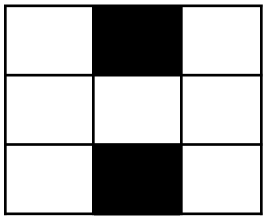
1. If solving cubie is on **Right**
   a. Execute: R Uw’ R Uw
2. If solving cubie is on **Left**
   a. Execute: L Uw’ L Uw
3. If solving cubie is on **Front**
   a. Execute: F Uw’ F Uw

4. If solving cubie is on **Back**
    a. Execute: B Uw’ B Uw
    b.
5. (Add Xw and Xw’ to data structure possible rotations)

j. Step 4

i. ff 2 cubies are in incorrect locations then swap them

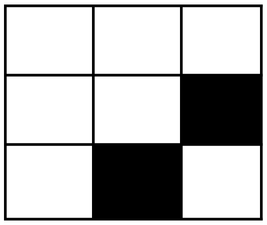
NOTE: Black squares on the below diagrams represents a white cubie on the down layer.

1. 
2. 

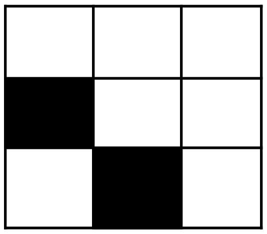
The above two cases can be swapped by rotating the up layer to match the alignment of 2, then executing the following:

M2, U2, M2

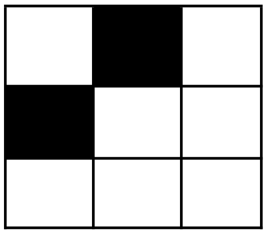
3.



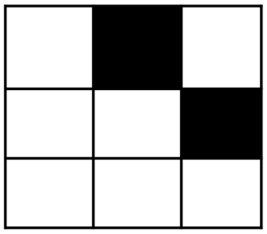
4.



5.



6.



7. To swap the white cubies in cases 3-6 above, rotate the down layer to match alignment of case 3. Then use the following algorithm:

   R2, U, F2, U’, R2

ii. If you have setup 3 above and they both need to be swapped and oriented, then you can use a single algorithm:
   1. F’ R F R2

3. F2L
4. OLL
5. PLL

You can use this website to generate an animation of the execution of the solution algorithm for those who don't know the rubik's cube notations:

- `https://ruwix.com/widget/3d/? [ solution algorithm ]`
- `This is helpful if my solution algorithm ends up being 80-100 movements`