Twisty Puzzle Masterz 2.0

(Project Notes)

Idea Conceived: 05/05/2023

- [ ] Create a really cool YouTube short to show the app working as soon as the app is done.
- [ ] Add detailed comments
    - explain all magic numbers and algorithms.
- [ ] Update variable and method names
- [ ] Use proper code style guide for java (reference software engineering document)

- [ ] Merge the opencv code into this project's code base. Keep the 2D cube interface code as a debugging option that can be changed in settings.

- [ ] Try to merge my code and the opencv code into the 2.0 repository
    - or try merging the opencv code into my project ?
    - Create a copy to test the merging?

- [ ] How to make a proper Java library for the solver feature
- [ ] Document all design decisions related to conventions that you follow for configuring the cube and its algorithms.
- [ ] Static code analysis tool for Java android in VC Code to generate code diagrams instead of having to make these by hand.

Working Java Android Cube Solver and OpenCV
https://github.com/ucchiee/AndroidRubikCubeSolver

How to add OpenCV 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=bR7lL886-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/

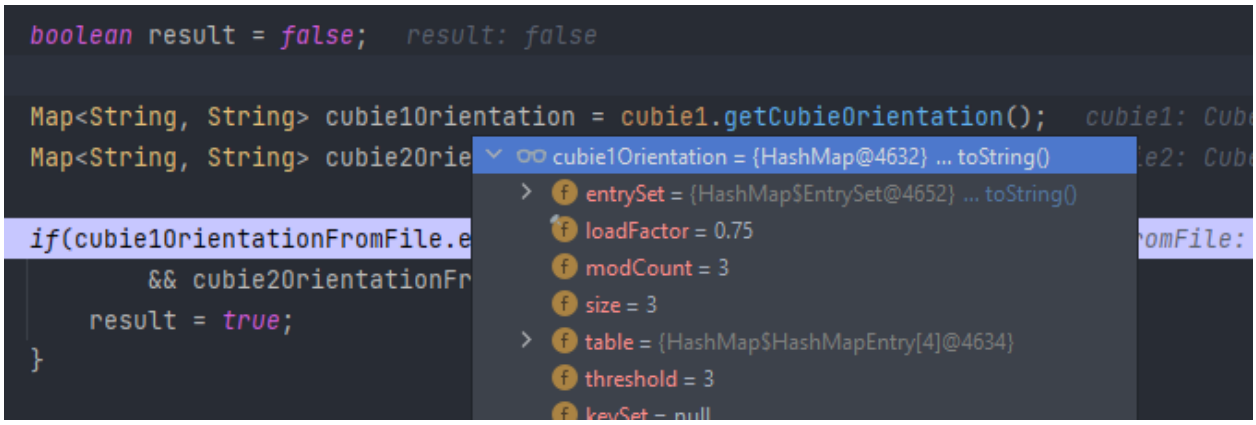
Video Format Inspiration:

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

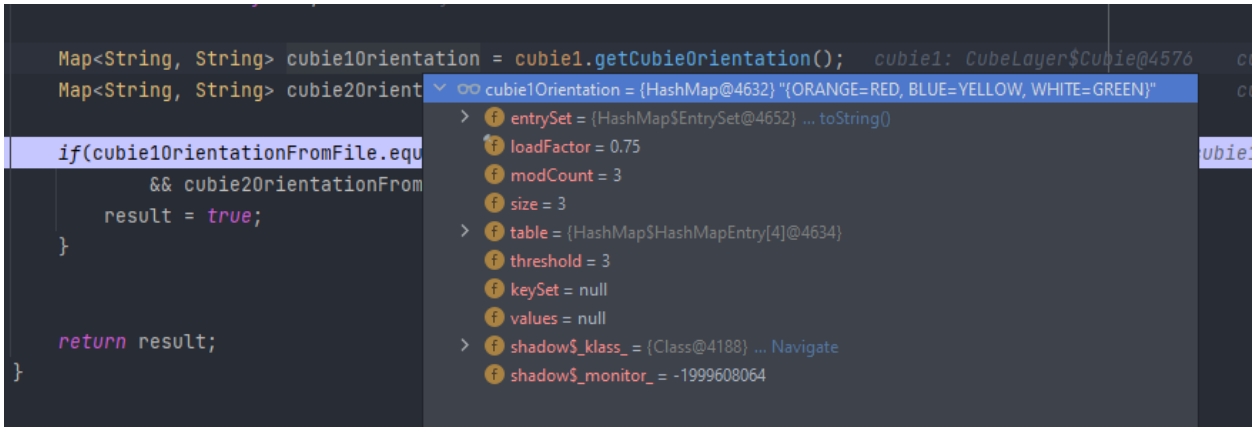
Debugging Error:

How to solve "Waiting until last debugger command completes"

- Fix 1
  https://stackoverflow.com/questions/63290304/how-to-solve-waiting-until-last-debugger-command-completes-stuck-in-android-st
- Fix 2
  - Open Task Manager -> Terminate all processes under Android Studio that are not Console Window Host



After turning off toString() object view to fix the above issue you'll need to select the toString() option from the drop down after hovering over a variable to see it's string value.

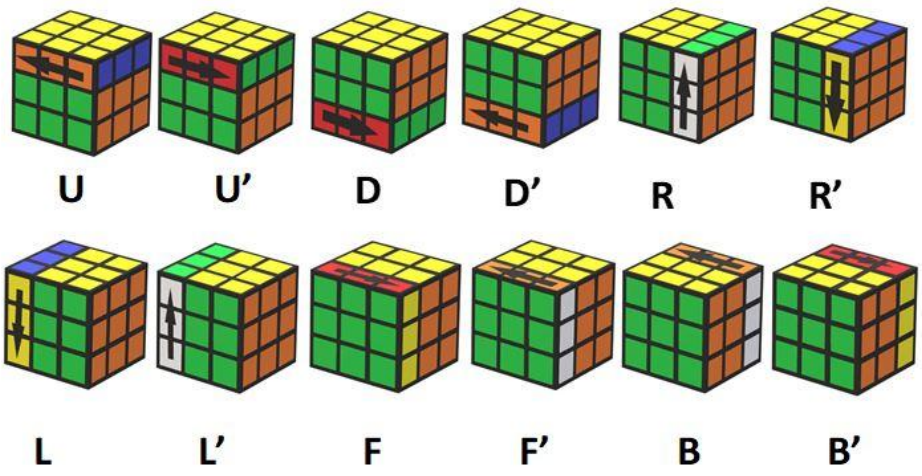


Terms

- ➢ Location:
  - All interesting layers for the given cubie
- ➢ Orientation
  - Placement of the stickers on the given cubie with respect to their correct center piece colors.
- ➢ Incorrectly Slotted
  - An edge cubie is in the wrong edge location
  - A corner cubie is in the wrong corner location
- ➢ Cubie
  - An individual piece on the cube.

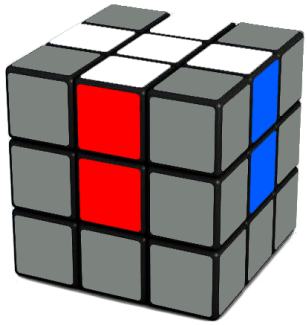
Notations

Layer Latter	Layer Name
U	Up
D	Down
R	Right
L	Left
F	Front
B	Back



How "*Twisty Puzzle Masterz*" solves the 3x3x3 Rubik's Cube

1. Cross



a. Solve Order

i. Cubies to solve

{

[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

Location = {U, R}	Execute => R2
Location = {U,F}	Execute => F2
Location = {U,L}	Execute => L2
Location = {U,B}	Execute => 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

Location = {L,F}	Execute => L or F'
Location = {R, F}	Execute => R' or F
Location = {L, B}	Execute => L' or B
Location = {R, B}	Execute => R or B'

Prefer the move that puts the cubie in the correct location.

(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 **avoidance maneuver**.

h. Step 2

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

i. Step 3

1. Set all cubies to their correct orientations.
2. If solving cubie is on **Right**
    a. R2 U F R' F'
3. If solving cubie is on **Front**
    a. F2 U L F' L'
4. If solving cubie is on **Left**
    a. L2 U B L' B'
5. If solving cubie is on **Back**
    a. B2 U R B' R'

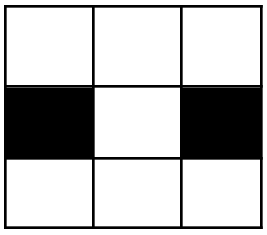
6. ***IMPROVEMENT 3:*** Add Xw and Xw' to data structure possible rotations
    7. If solving cubie is on **Right**
        a. Execute: R Uw' R Uw
    b. If solving cubie is on **Front**
        i. Execute: F Uw' F Uw
    8. If solving cubie is on **Left**
        a. Execute: L Uw' L Uw
    9. If solving cubie is on **Back**
        a. Execute: B Uw' B Uw

j. Step 4

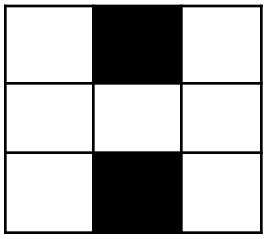
i. Iff 2 cubies are in incorrect locations then swap them

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

1.



2.



ii. Case 1:

R2 L2 U2 R2 L2

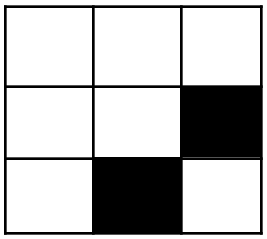
iii. Case 2:

F2 B2 U2 F2 B2

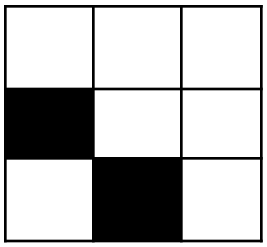
1. ***IMPROVEMENT 4:*** Improve the rubik's cube solution code and data structure to be able to handle whole cube rotations.
2. ***IMPROVEMENT 5***: The above two cases can be swapped by rotating the up layer to match the alignment of 2, then executing the following:

M2, U2, M2

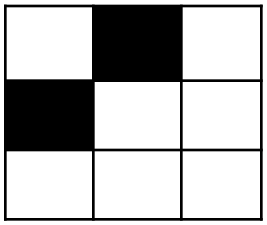
3.



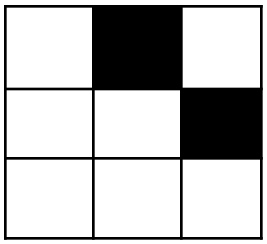
4.



5.



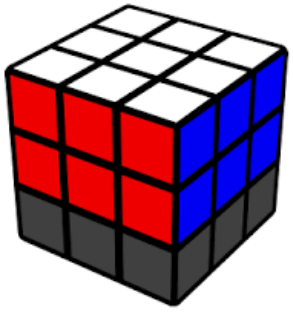
6.



7. To swap the white cubies in cases 3-6 above, use these four algorithms respectively:
    a. R2, U, F2, U', R2 // [[D,F] , [D,R]]
    b. F2, U, L2, U', F2 // [[D,F] , [D,L]]
    c. L2, U, B2, U', L2 // [[B,D] , [D,L]]
    d. B2, U, R2, U', B2 // [[B,D] , [D,R]]

iv. ***IMPROVEMENT 6***: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

2. F2L



a. F2L Configuration Format:

General

```
"Case Number": {
                <Cubie1>
              , <Cubie2>
              , <Solution Algorithm>
  }
```

Specific

```
{
      "Case": 1,
      "Cubie1": {
            "Orientation": {
              "STICKER_COLOR_1": "SIDE_COLOR_1",
              "STICKER_COLOR_2": "SIDE_COLOR_2",
              "STICKER_COLOR_3": "SIDE_COLOR_3"
            }
      },
      "Cubie2": {
            "Orientation": {
              "STICKER_COLOR_3": "SIDE_COLOR_2",
              "STICKER_COLOR_2": "SIDE_COLOR_3"
            }
      },
      "SolutionAlgorithm": "F' U F' U2 R' F2 R U2 F2"
}
```

F2L Configuration File: Either use JSON or XML file to store all 41 F2L algorithms. *(This same type of file will be needed for OLL and PLL)*

- XML
  - https://www.geeksforgeeks.org/xml-parsing-in-android-using-dom-parser/
- JSON
  - https://www.geeksforgeeks.org/json-parsing-in-android/

Cube Orientation = { {F->BLUE}, {U->YELLOW}, {R->RED} }

b. **F2L Edge Cases** (that require a specific correction algorithm)
   i. White stickered Cubie is in the D layer in the wrong location.
   ii. White stickered Cubie is in the D layer in the correct location with wrong orientation.
   iii. White stickered Cubie is in the D layer in the correct location and correct orientation.
   iv. F2L edge cubie is in the wrong location in the middle layer.

c. **F2L Solution Steps:**
   i. Locate both cubies
   ii. If one or both of the F2L cubies are slotted in the wrong corner, then remove them from that corner. (***removal maneuver***)
   iii. Rotate the top layer to correctly align the F2L pair iff:
      1. Both of the F2L cubies are in the U layer.
      2. One F2L cubie is in the U layer and the other is in the correct location.

   iv. Execute the corresponding algorithm using the (***orientation transform***) function.

3. Pseudo-Code:

```
// Pairs to solve, in order
{ [ORANGE, WHITE, BLUE], [ORANGE, BLUE] }
{ [RED, WHITE, BLUE], [RED, BLUE] }
{ [RED, WHITE, GREEN], [RED, GREEN] }
{ [ORANGE, WHITE, GREEN], [ORANGE, GREEN] }
```

```
      List<SurfaceName> locationOfCubie1 = findLocationOfCubie( cubie1 );
      List<SurfaceName> locationOfCubie2 = findLocationOfCubie( cubie2 );

List<SurfaceName> correctLocation1 = getCubieAtLocation( String[] {“F R”} );
List<SurfaceName> correctLocation2 = getCubieAtLocation( String[] {“F R D”});


            if( locationOfCubie1 != correctLocation1
                                    && locationOfCubie2 != correctLocation2
                                 ){
                  if( isInWrongEdge( cubie1  ) ){
                        // Removal maneuver       (for edges)
                  }
                  if( isInWrongCorner( cubie2 )){
                        // Removal maneuver        (for corners)
                  }

                  // Loop through all F2L Cases to find the one that matches.
                  String correctF2LCase = {};
                  for(String case : F2L_JSON ){
                        // Check orientation and location of cubie1 and 2
                  with that of the current
                        //      F2L case.

                        if( foundMatchingCase ){
                              break;
                        }
                  }

                  String algorithm = correctF2LCase [“SolutionAlgorithm”];
                  String algorithmTransform = orientationTransform( algorithm
                  );

            }
```

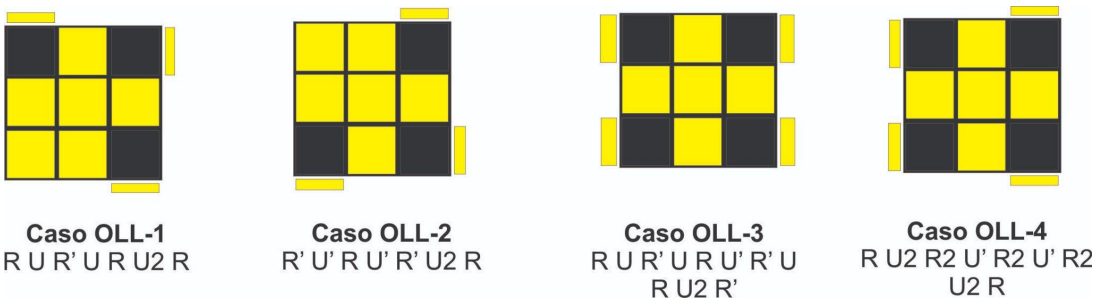
4. OLL

1. Now that you have solved the cross, and the first 2 layers (F2L), the next step is to solve the orientation of the last layer (OLL)

a. OLL Configuration Format:

```
"Case 1": {
        "IsYellowMap_TopSurface"
             [    F,T,F,
                  T,T,T,
                  T,T,F  ]
        "IsYellowMap_Borders"
             [T,F,F], [T,F,F], [T,F,F], [F,F,F]

        "SolutionAlgorithm" =
                  "R U R' U R U2 R"
  }
```



Caso OLL-1
R U R' U R U2 R

Caso OLL-2
R' U' R U' R' U2 R

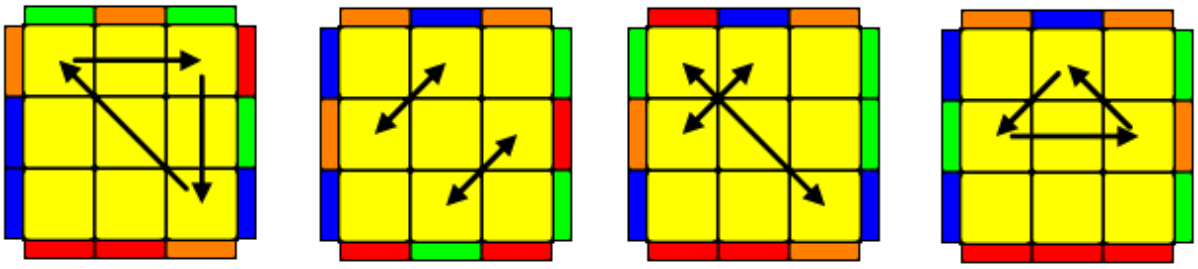
Caso OLL-3
R U R' U R U' R' U
R U2 R'

Caso OLL-4
R U2 R2 U' R2 U' R2
U2 R

b. **OLL Solution Steps:**

c. **<u>OLL Pseudo-Code</u>**

5. PLL



Other Helpful Resources

1. 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:
    a. https://ruwix.com/widget/3d/? [solution algorithm]
    b. This is helpful if my solution algorithm ends up being 80-100 movements

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

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

4. https://www.youtube.com/watch?v=afAGtExoiLQ
5. https://www.youtube.com/watch?v=RMo_CLi1Z5g
6. https://www.youtube.com/watch?v=CWmKHcx1X6A
7. https://www.youtube.com/watch?v=3pqo6SMmtS4
8. Project is 9 years old. Can't build into Android Project without Gradle.
    a. https://sgelb.github.io/projects/arcs
9. 3D cube animation
    a. https://github.com/cjurjiu/AnimCubeAndroid
    b.