

# 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>

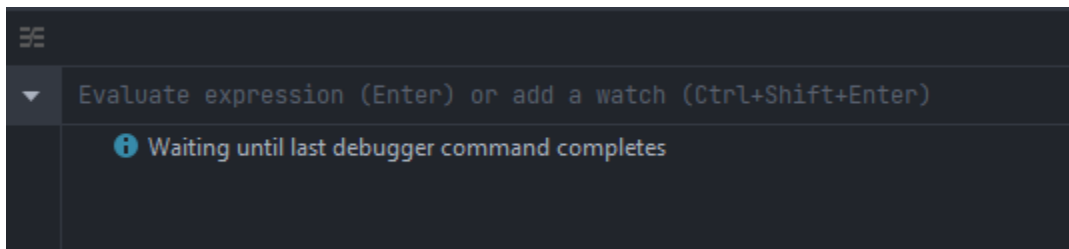
### How to add OpenCV to android:

- <https://www.youtube.com/watch?v=olk2hTPxFqs&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/>

#### Video Format Inspiration:

[https://www.youtube.com/watch?v=fAX27\\_FyU9g](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-studio/>

- Fix 2
  - Open Task Manager -> Terminate all processes under Android Studio that are not Console Window Host

```

boolean result = false;  result: false

Map<String, String> cubie1Orientation = cubie1.getCubieOrientation();  cubie1: Cube
Map<String, String> cubie2Orientation = cubie2.getCubieOrientation();  cubie2: Cube

if(cubie1OrientationFromFile.equals(cubie2OrientationFromFile)
    && cubie2OrientationFromFile != null)
    result = true;
}

```

Tooltip for `cubie1Orientation`:

```

cubie1Orientation = {HashMap@4632} ... toString()
  > f entrySet = {HashMap$EntrySet@4652} ... toString()
  f loadFactor = 0.75
  f modCount = 3
  f size = 3
  > f table = {HashMap$HashMapEntry[4]@4634}
  f threshold = 3
  f keySet = null

```

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.

```

Map<String, String> cubie1Orientation = cubie1.getCubieOrientation();  cubie1: CubeLayer$Cube@4576
Map<String, String> cubie2Orientation = cubie2.getCubieOrientation();  cubie2: CubeLayer$Cube@4576

if(cubie1OrientationFromFile.equals(cubie2OrientationFromFile)
    && cubie2OrientationFromFile != null)
    result = true;
}

return result;
}

```

Tooltip for `cubie1Orientation`:

```

cubie1Orientation = {HashMap@4632} "{ORANGE=RED, BLUE=YELLOW, WHITE=GREEN}"
  > f entrySet = {HashMap$EntrySet@4652} ... toString()
  f loadFactor = 0.75
  f modCount = 3
  f size = 3
  > f table = {HashMap$HashMapEntry[4]@4634}
  f threshold = 3
  f keySet = null
  f values = null
  > f shadow$_klass_ = {Class@4188} ... Navigate
  f shadow$_monitor_ = -1999608064

```

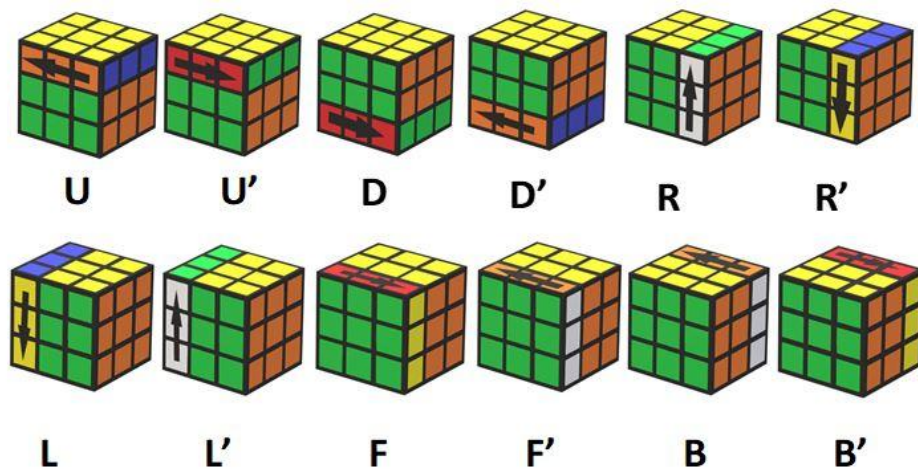
## Terms

- 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



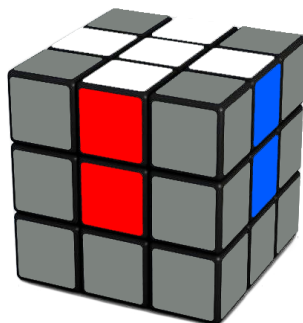
## Substitute Notations

- $x = R w' L$                        $x' = R w L'$
- $y = U w D'$                          $y' = U w' D$
- $z = B w' F$                           $z' = B w F'$
- $s = B w' B$                           $s' = B w B'$

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 intersects with the down and chosen layer to move.

// Chosen layer to move  
 $n = [L \text{ or } R]$

- D (until no white edge cubie intersects  $[n, D]$  )
- $n$  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 won't 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  $X_w$  and  $X_w'$  to data structure possible rotations

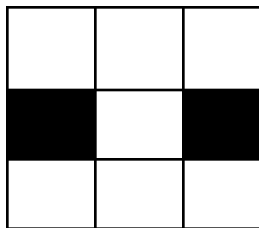
- If solving cubie is on **Right**
  - Execute:  $R U_w' R U_w$
- If solving cubie is on **Front**
  - Execute:  $F U_w' F U_w$
- If solving cubie is on **Left**
  - Execute:  $L U_w' L U_w$
- If solving cubie is on **Back**
  - Execute:  $B U_w' B U_w$

j. Step 4

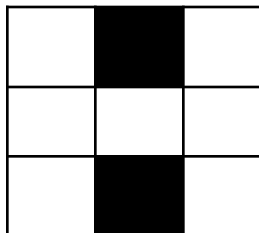
- i. If 2 cubies are in incorrect locations then swap them

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

1.



2.



- ii. Case 1:

$R^2 L^2 U^2 R^2 L^2$

- iii. Case 2:

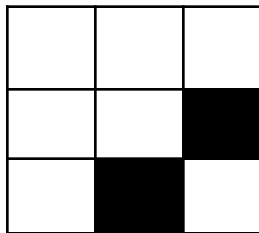
$F^2 B^2 U^2 F^2 B^2$



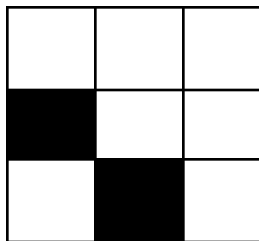
1. **IMPROVEMENT 4:** Improve the rubik's cube solution code and data structure to be able to handle whole cube rotations.
2. **IMPROVEMENT 5:** The above two cases can be swapped by rotating the up layer to match the alignment of 2, then executing the following:

- M2, U2, M2

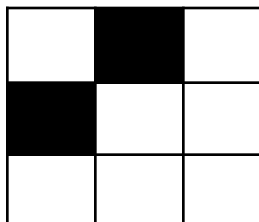
3.



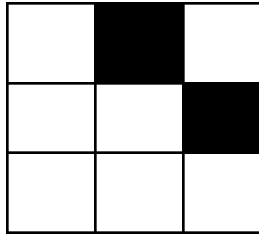
4.



5.



6.



7. To swap the white cubies in cases 3-6 above, use these four algorithms respectively:

- |                       |                      |
|-----------------------|----------------------|
| a. $R^2 U F^2 U' R^2$ | // [ [D,F] , [D,R] ] |
| b. $F^2 U L^2 U' F^2$ | // [ [D,F] , [D,L] ] |
| c. $L^2 U B^2 U' L^2$ | // [ [B,D] , [D,L] ] |
| d. $B^2 U R^2 U' B^2$ | // [ [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.  $F' R F R^2$

## 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

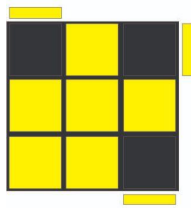


1. Now that you have solved the cross, and the first 2 layers (F2L), the next step is to solve the orientation of the last layer (OLL)

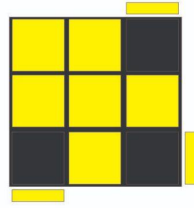
a. **OLL Configuration Format:**

This is the format that the **OLL\_algorithms.json** file is expected to use.

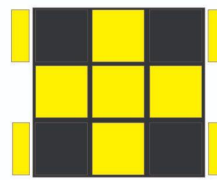
```
{
  "Case": 1,
  "YellowMap_TopSurface": [
    "" , "Y" , "",
    "" , "Y", "" , "Y",  "",
    "Y", "" , "Y", "Y",  "",
    "" , "Y", "Y", "Y",  "",
    "" , "" , ""
  ],
  "SolutionAlgorithm" = "R U R' U R U2 R"
}
```



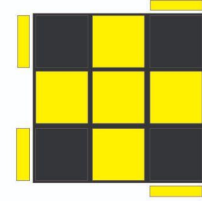
**Caso OLL-1**  
R U R' U R U2 R



**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:

### 1. Paper Orientation:

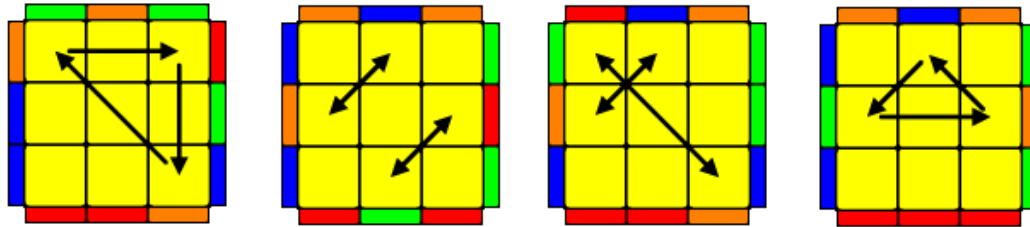
- Loop through each of the 57 cases to check if you have any of them
- If not, then perform a U rotation and check all 57 again
- Once you've found a case that matches, with the correct orientation, then execute the associated algorithm

### i. **IMPROVEMENT 7:**

- Instead of checking for each of the 57 cases and doing a U rotation until the correct alignment is made, you could just create a kind of transformOrientation() for OLL to execute a transformed version of the solution algorithm, despite the current orientation.

## c. OLL Pseudo-Code

## 5. PLL



#### a. OLL Configuration Format:

```

“Case 1” : {
    “MoveTo”:
        [ “U R B”,  self , “U R F” ,
          self      , self , self      ,
          self      , self , “U L B” ,
            ]
    “SolutionAlgorithm” : “x R’ U R’ D2 R U’ R’ D2 R2”
}

```

The above case format show that top:

Cubie at location “U L B” is moving to location “U R B”,  
 Cubie at location “U R B” is moving to location “U R F”,  
 Cubie at location “U R F” is moving to location “U L B”

#### Locations On Up Layer

```

[
    “U L B” , “U B” , “U R B” ,
    “U L”   , “U”   , “U R”   ,
    “U L F” , “U F” , “U R F”
]

```



b. **OLL Solution Steps:**

- i. Loop through each of the 21 cases to check if you have any of them
- ii. If not, then perform a U rotation and check all 21 again
- iii. Once you've found a case that matches, with the correct orientation, then execute the associated algorithm

c. **OLL Pseudo-Code**

## 6. Algorithm Errors

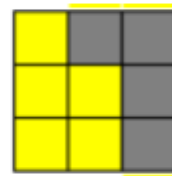
- i. All of the algorithms listed in this section on the left are used in the oll.pdf document in the /notes directory of this android project. After testing all 57 oll algorithms, I found these to be incorrect. Their correct replacements are listed on the right.

### Wrong

31

$(R' U') F (U R U' R') F R$

### Correct Replacement

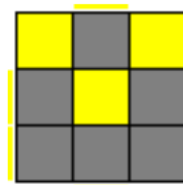


(same case, different orientation)

$L' D w' R D w L U' R w' U' R w$

19

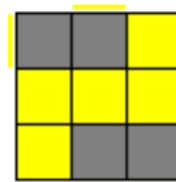
$R w' (R U) (R U R' U' R w) \times (R^2 U) (R U')$



$R' U^2 F R U R' U' U w' D R^2 U^2 R B$

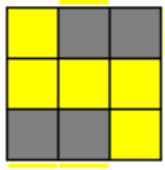
39

$(L F') (L' U' L U) F U' L'$

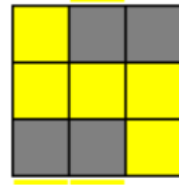


$R B' R' U' R U U w D' R U' F'$

40



$(R' F) (R U R'$   
 $U') F' U R$



$L' B L U L' U' U w' D L' U F$

## 7. Test Cases

a. Test the following initial Rubik's Cube rotations on a solved cube to ensure that:

i. 'w' rotations are working correctly.

1. Rw
2. Rw'
3. Lw
4. Lw'
5. Uw
6. Uw'
7. Dw
8. Dw'
9. Fw
10. Fw'
11. Bw
12. Bw'
13. Fw R U R' U' Fw'
14. Fw Rw'

ii. 'M' rotations are working correctly

1. M
2. M'
3. M2
4. M2'
5. M' U M U2 M' U M
6. F' U' R' M U R B'
- 7.

b.

## 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](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.