**Yuxuan’s 4 Look Last Layer Tutorial**

**This document attempts to explain how to do 4 look last layer for 3x3. It assumes that you know how to read notation.**

**Note 1:** The pictures will be presented in the angle you are supposed to hold the cube at to perform the algorithm.

**Note 2:** All algorithms will also be broken down into triggers (a sequence of moves that is fast and easy to execute). Each case will have two cells in the algorithm section. The first cell will have the normal algorithm(s). The second cell will have the same algorithm(s) broken down into triggers.

**Note 3:** For the most part, all algorithms will be written the way I execute them (some double turns will have a ‘ even though a 180 degree turn is the same whether the layer was turned clockwise or counter clockwise). This is to help you fingertrick the algorithm.

**Note 4:** My philosophy for learning algorithms was/is picking algorithms that are easy to learn (even if it might mean it is more moves/slower) so many of my algorithms build on each other.

**Note 5:** If there are any problems with the algorithms, contact Yuxuan.

This guide assumes that you know how to solve the first two layers (F2L) and begins right after the first two layers are solved. In other words, this guide is for last layer (LL) only.

**4LLL is split into 2 look OLL and 2 look PLL**

**2 Look OLL**

**Step 1. Orient Edges**

**(This is the same as the Edge Orientation step in the MCC Beginner Method)**

**Orient the edges of the last layer by using FRURUF from the correct angle.**

**There are 4 possible cases (one of them is solved/skip).**

|  |  |
| --- | --- |
| **Case 1**  **F R U R’ U’ F’** | **Case 2**  **F R U R’ U’ R U R’ U’ F’ (Case 1 twice)** |
| **Case 3**  **F R U R’ U’ F’  U2**  **F R U R’ U’ R U R’ U’ F’**  **(Case 1 + Case 2)** | **Case 4**  **Solved** |

**Step 2. Orient Corners**

**There are 8 cases (1 of them is solved/skip)**

**All edges oriented**

|  |  |  |
| --- | --- | --- |
| **OLL** | **Algorithm(s)** | **Comments** |
|  | R U R’ U R U2’ R’ | “Sune” |
| With triggers:  (R U R’ U) (R U2’ R’) |
|  | R’ U’ R U’ R’ U2’ R | “Antisune” |
| With triggers:  (R’ U’ R U’) (R’ U2’ R) |
|  | 1. R U2’ R’ U’ R U R’ U’ R U’ R’  2. F (R U R’ U’)3 F’  (expanded: F R U R’ U’ R U R’ U’ R U R’ U’ F’) | “Double Sune”  1. Happens to solve corners if  UBR == UFR and UBL == UFL  2. Solves corners if UBL == UBR and UFL == UFR |
| With triggers:  1. (R U2’ R’) U’ (R U R’ U’) (R U’ R’)  2. F (R U R’ U’)3 F’ |
|  | R U2’ R2’ U’ R2 U’ R2’ U2’ R | “Pi”  Solves corners if UBR == UFR (and UBL and UFL are opposite colors) |
| With triggers:  (R U2’) (R2’ U’) (R2 U’) (R2’ U2’) R |
|  | 1. R2 D’ R U2’ R’ D R U2 R  2. L2’ D L’ U2 L D’ L’ U2 L’  3. (U2) R U R’ U R U2’ R2’ U’ R U’ R’ U2’ R | “Headlights”, “U”  1. Solves corners if UBL == FUR  2. Solves corners if UBR == FUL  3. Solves corners if UFL and UFR are opposite colors and the back 2 corners are solved. Algorithm is basically  sune + antisune |
| With triggers:  1. R2 D’ (R U2’ R’) D (R U2 R)  2. L2’ D (L’ U2 L) D’ (L’ U2 L’)  3. (U2) (R U R’ U) (R U2’ R’) (R’ U’ R U’) (R’ U2’ R) |
|  | r U R’ U’ r’ F R F’ | “Chameleon”  Very similar to the case below |
| With triggers:  (r U R’ U’) (r’ F R F’) |
|  | F’ r U R’ U’ r’ F R | “Triple Sune”, “Bowtie”  Very similar to the case above (move the last move to the front) |
| With triggers:  F’ (r U R’ U’) (r’ F R) |

**2 Look PLL**

**Step 1: Solve Corners**

**Look at the corners. If they are solved relative to each other, skip to the next step.**

**If they are not solved relative to each other, see if the corners need an adjacent or diagonal swap.**

**Corners Only**

| **PLL** | **Algorithm(s)** | **Comments** |
| --- | --- | --- |
|  | R U R' U' R' F R2 U' R' U' R U R' F' | T perm  Look for headlights, place on left |
| With triggers:  (R U R' U') (R' F) (R2 U' R' U') (R U R' F') |
|  | F R U' R' U' R U R' F' R U R' U' R' F R F' | Y perm |
| With triggers:  F (R U' R' U') (R U R' F') (R U R' U') (R' F R F') |

**Step 2: Solve edges**

|  |  |  |
| --- | --- | --- |
| **PLL** | **Algorithm(s)** | **Comments** |
|  | M2’ U’ M2’ U2’ M2’ U’ M2’ | H perm |
| With triggers:  (M2’ U’ M2’) U2’ (M2’ U’ M2’) |
|  | M2 U’ M2’ U’ M’ U2’ M2’ U2’ M’ U2’ | Z perm |
| With triggers:  (M2 U’) (M2’ U’) (M’ U2’) (M2’ U2’) (M’ U2’) |
|  | R U' R U R U R U' R' U' R2 | Ua perm  Not opposite color on left  Opposite color on right |
| With triggers:  (R U' R U) (R U R U') (R' U' R2) |
|  | R2 U R U R' U' R' U' R' U R' | Ub perm  Opposite color on left  Not opposite color on right  Inverse of Ua perm |
| With triggers:  (R2 U) (R U R' U') (R' U' R' U) R' |