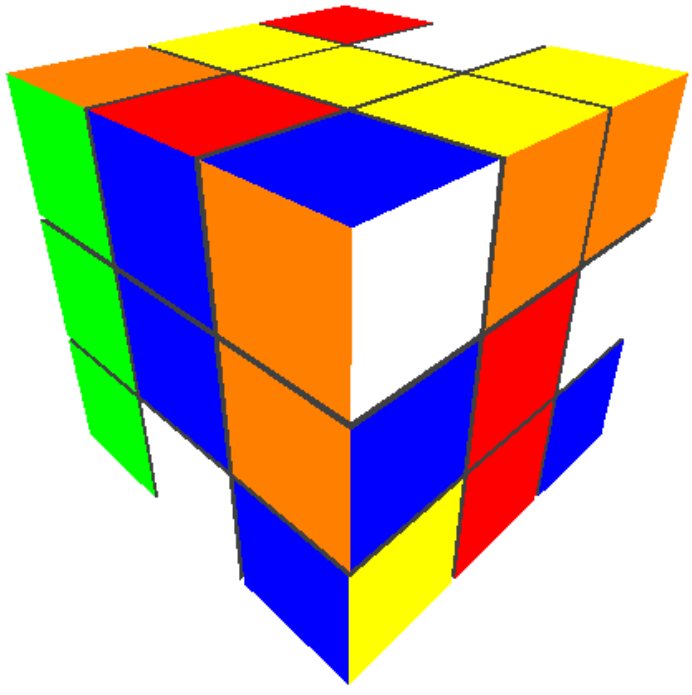


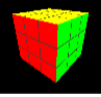
Rubik's Cube in OpenGL

Felix Baumann, Jonas Winkler,
Ravinder Sangar

Einführung Computergraphik PS

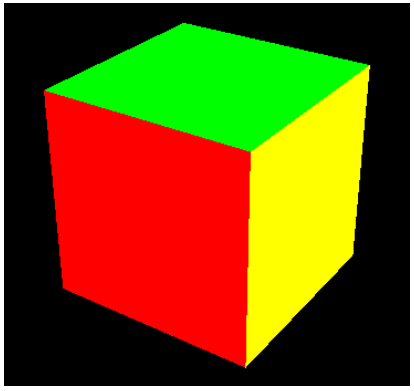
25.06.2021

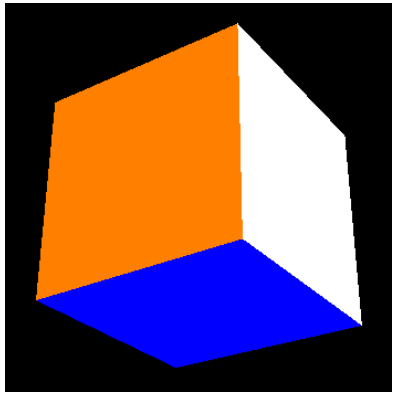
Übersicht



Erzeugen des Würfels

- Gesamte Rubikscube besteht aus 27 (26) Würfel
- Dabei gilt:
  - Rot gegenüber Orange
  - Weiß gegenüber Gelb
  - Grün gegenüber Blau



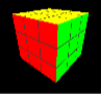


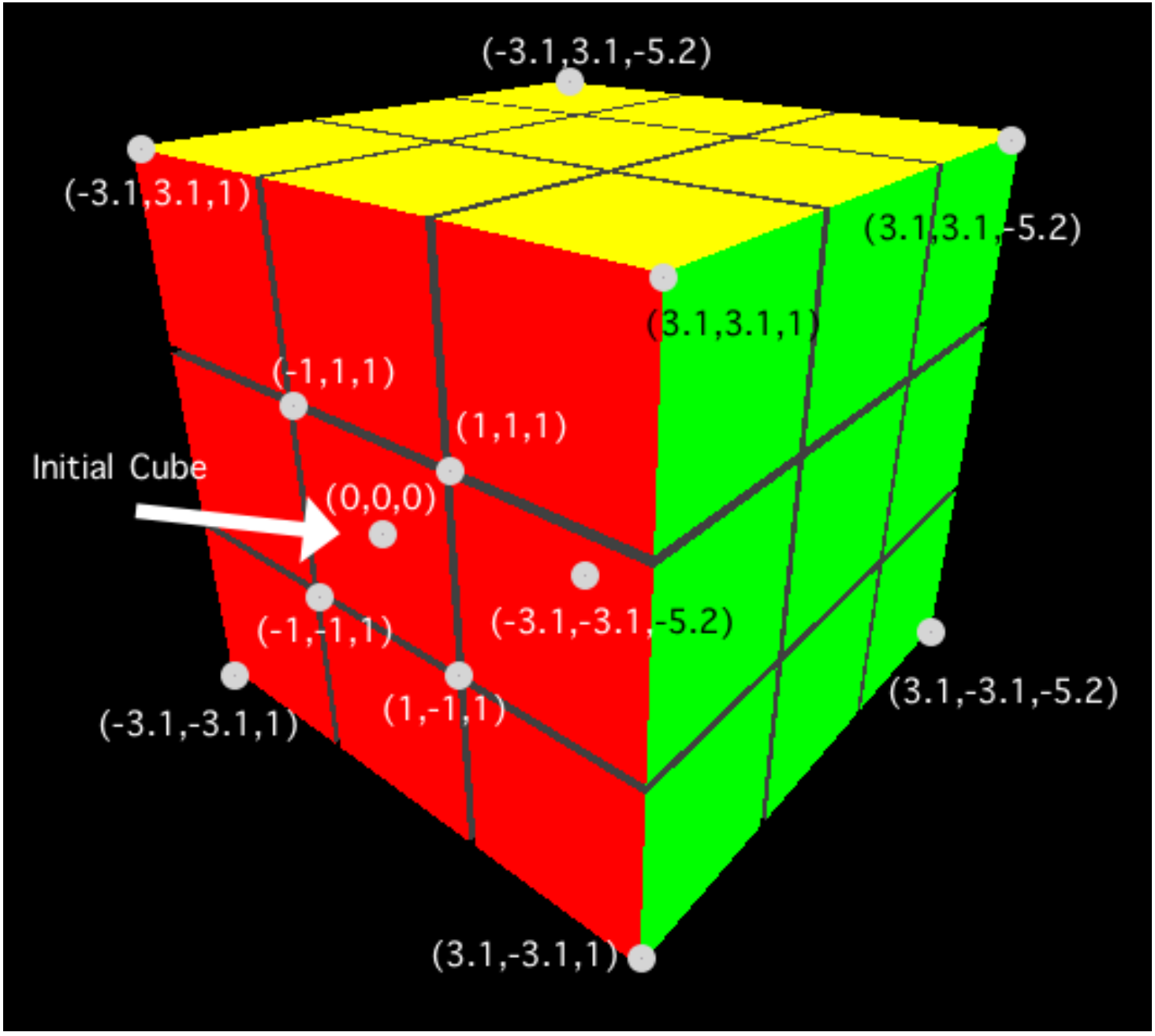
Erzeugen des Würfels

- 27 Objekte der Cube-Klasse
- Initialwürfel als Array mit Koordinaten (x,y,z) und Farben (r,g,b)
- Alle anderen bauen auf diesem auf
- Positionen (x,y,z) werden angepasst
- Farben bleiben gleich

```
GLfloat initCube[6*36] = {
    -1.0f,-1.0f,-1.0f,   1.0f, 1.0f, 1.0f, /* 0-35 bottom: white */
     1.0f,-1.0f,-1.0f,   1.0f, 1.0f, 1.0f,
    -1.0f,-1.0f, 1.0f,   1.0f, 1.0f, 1.0f,
     1.0f,-1.0f,-1.0f,   1.0f, 1.0f, 1.0f,
     1.0f,-1.0f, 1.0f,   1.0f, 1.0f, 1.0f,
    -1.0f,-1.0f, 1.0f,   1.0f, 1.0f, 1.0f,
```

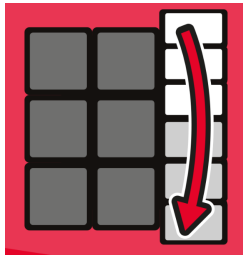
```
static Cube cube[] ={
    Cube(MIDDLE, 0.0f),
    Cube(LEFT, 0.0f),
    Cube(RIGHT, 0.0f),
    Cube(TOP, 0.0f),
    Cube(BOTTOM, 0.0f),
    Cube(TOP_LEFT, 0.0f),
    Cube(TOP_RIGHT, 0.0f),
    Cube(BOTTOM_LEFT, 0.0f),
    Cube(BOTTOM_RIGHT, 0.0f),
    Cube(MIDDLE, -2.1f),
    Cube(LEFT, -2.1f),
    Cube(RIGHT, -2.1f),
    Cube(TOP, -2.1f),
    Cube(BOTTOM, -2.1f),
    Cube(TOP_LEFT, -2.1f),
    Cube(TOP_RIGHT, -2.1f),
    Cube(BOTTOM_LEFT, -2.1f),
    Cube(BOTTOM_RIGHT, -2.1f),
    Cube(MIDDLE, -4.2f),
    Cube(LEFT, -4.2f),
    Cube(RIGHT, -4.2f),
    Cube(TOP, -4.2f),
    Cube(BOTTOM, -4.2f),
    Cube(TOP_LEFT, -4.2f),
    Cube(TOP_RIGHT, -4.2f),
    Cube(BOTTOM_LEFT, -4.2f),
    Cube(BOTTOM_RIGHT, -4.2f),
};
```



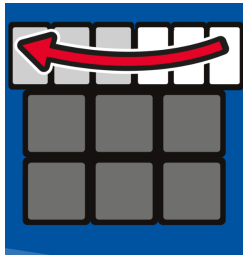

(-3.1,3.1,-5.2)
(-3.1,3.1,1)
(3.1,3.1,-5.2)
(3.1,3.1,1)
(-1,1,1)
(1,1,1)
Initial Cube
(0,0,0)
(-1,-1,1)
(-3.1,-3.1,-5.2)
(-3.1,-3.1,1)
(1,-1,1)
(3.1,-3.1,-5.2)
(3.1,-3.1,1)

3 Rotationen

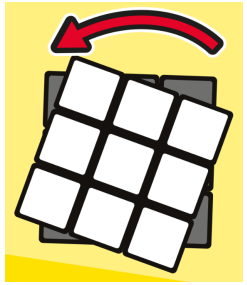
- Rotation um die X-Achse



- Rotation um die Y-Achse



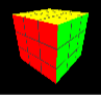
- Rotation um die z-Achse



- Alle Rotationen jeweils um 90°

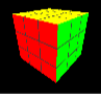
[1]

[1] https://www.youcandothecube.com/solve-it/3x3-solution



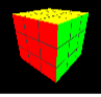
Rotation um X-Achse

- glm::translate(glm::vec(0, 0, 2.1))
- glm::rotate(glm::vec(1, 0, 0))
- glm::translate(- glm::vec(0, 0, 2.1))



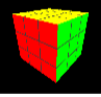
Rotation um X-Achse

- glm::translate(glm::vec(0, 0, 2.1))
- glm::rotate(glm::vec(1, 0, 0))
- glm::translate(- glm::vec(0, 0, 2.1))



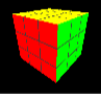
Rotation um y- und z-Achse

- Für y-Achse im Grunde Analog
- Translation $\rightarrow$ Rotation $\rightarrow$ Translation aufheben

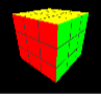


Rotation um y- und z-Achse

- Für y-Achse im Grunde analog
- Translation $\rightarrow$ Rotation $\rightarrow$ Translation aufheben
- Für die z-Achse zunächst keine Translation nötig

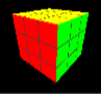


Rotation Erfolgreich ... Aber



Rotationsproblem

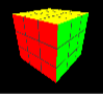
- Der gesamte Würfel rotierte um die jeweilige Achse
- Die richtigen Ebenen rotierten, aber der Würfel setzte sich zurück



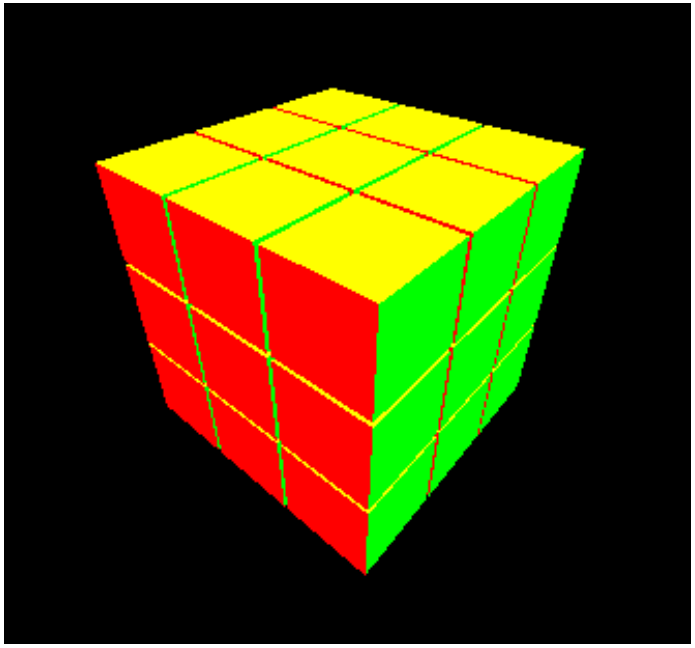
Rotationsproblem

- Jede Rotation berechnet eine Matrix aber auf was wenden wir diese an?
- Jeder Teilwürfel benötigt sein eigenes Koordinatensystem
- Beim Zeichnen wird pro Teilwürfel die eigene Matrix verwendet
- Rotation $\rightarrow$ neue Matrix

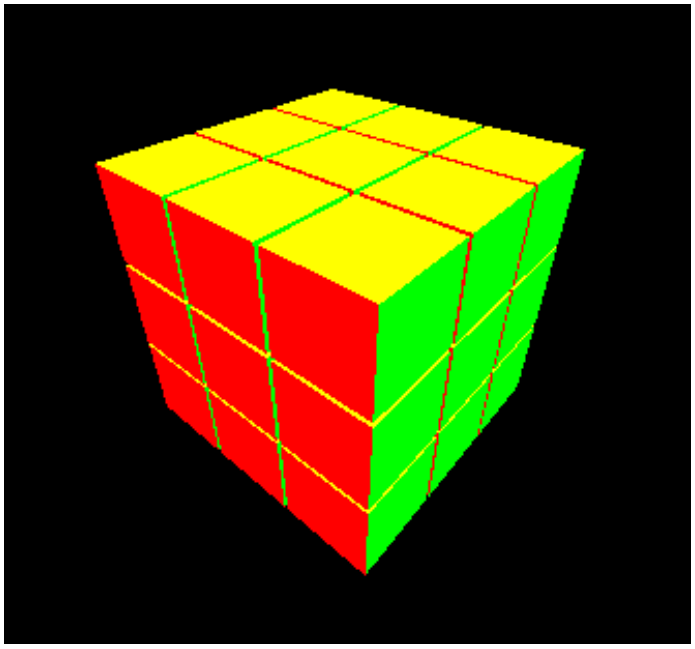
Was, wenn wir nun mehrere Rotationen nacheinander ausführen?

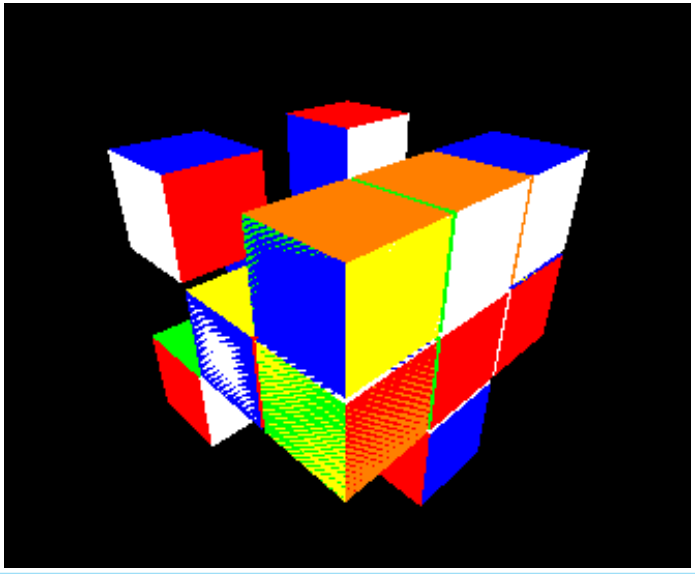


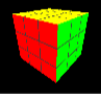
Achsenproblem



Achsenproblem



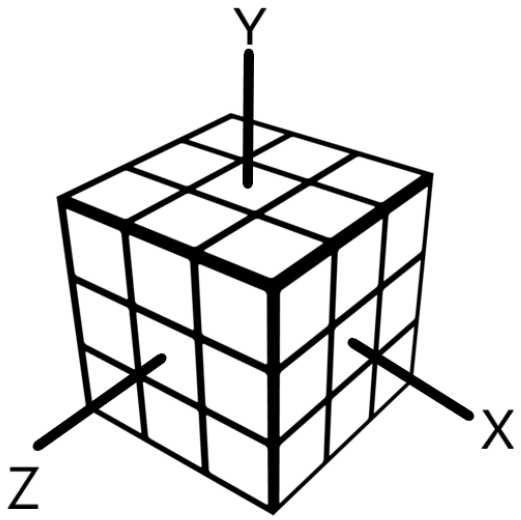




Achsenproblem

Initialzustand vom Würfel

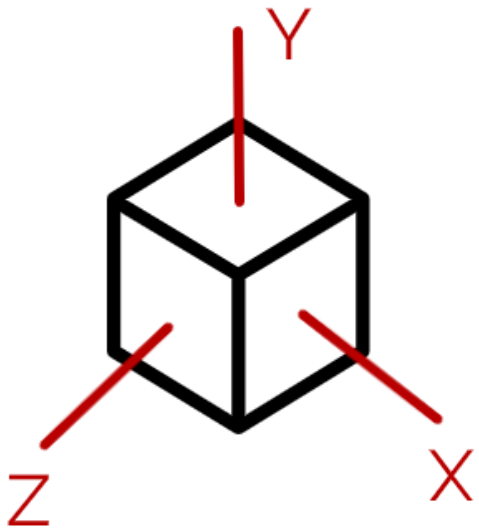
Rotation um X-Achse



Achsenproblem

Initialzustand vom Würfel

Rotation um X-Achse

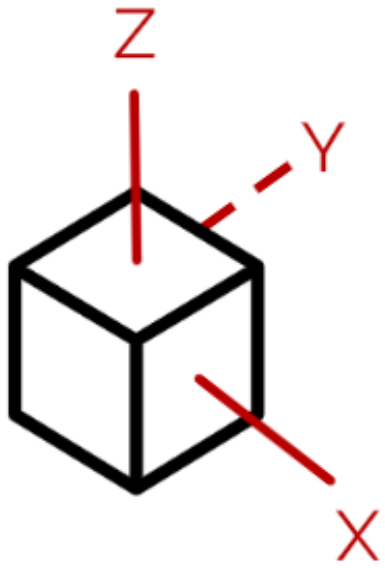


Achsenproblem

Initialzustand vom Würfel

Rotation um x-Achse

y- und z-Achse vertauscht
y-Richtung geändert

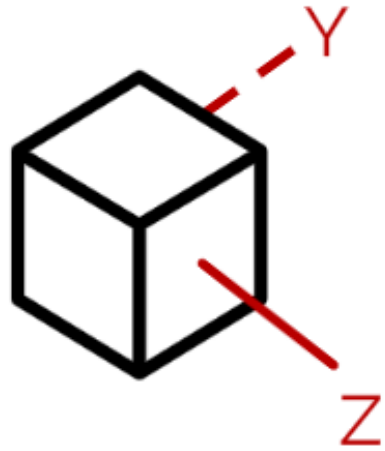


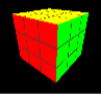
Achsenproblem

Initialzustand vom Würfel

Rotation um ursprüngliche Z-Achse

X- und z-Achse vertauscht
X-Richtung geändert



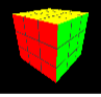


Einbindung des Lösungsalgorithmus

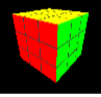
- separate Klasse für Algorithmus: AlgoCube
- randomisiert Würfel $\rightarrow$ speichert in Vektor randomizeCubeMoves
- löst diesen mittels Layer-By-Layer-Methode $\rightarrow$ speichert in Vektor solverMoves
- Objekt wird im Main-File erzeugt, Vektoren ausgelesen und abgearbeitet

Struktur von AlgoCube-Klasse

- 3*3*3 Array von CubePieces mit Farben als Characters, hierarchisch angeordnet
- Implementierung von 24 Moves
  - 6 für Rotation des ganzen Würfels
  - 18 für Rotation der einzelnen Ebenen (6 pro Achse)
- Ausführung eines Moves ändert nicht nur Position, sondern auch Anordnung $\rightarrow$ muss an Hierarchie angepasst werden

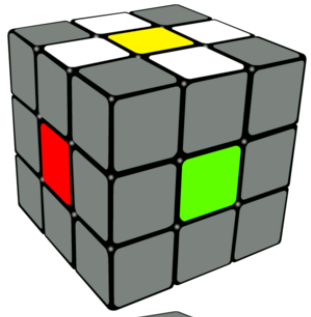


Layer-By-Layer-Algorithmus

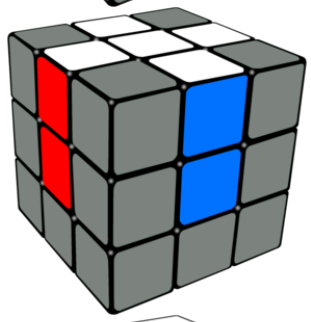


Lösungsalgorithmus in 8 Schritten - Untere Ebene

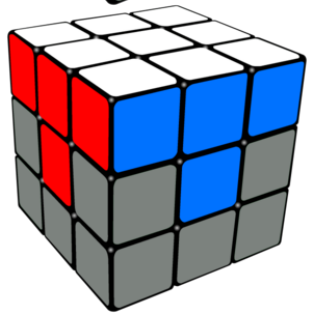
1. Gänseblümchen



2. Weißes Kreuz



3. Weiße Ecken

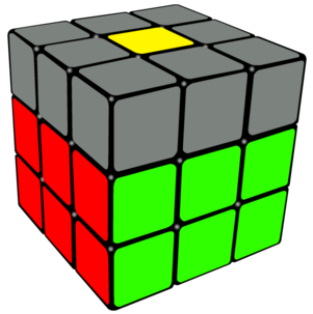


2

[2] http://www.learnhowtosolvearubikscube.com/how-to-solve-a-rubiks-cube-solution-overview

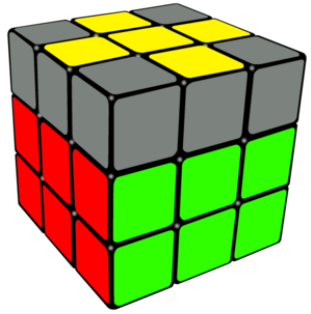
Lösungsalgorithmus in 8 Schritten - Mittlere Ebene

1. Kanten in mittlerer Ebene

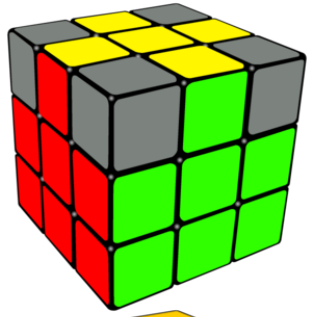


Lösungsalgorithmus in 8 Schritten - Obere Ebene

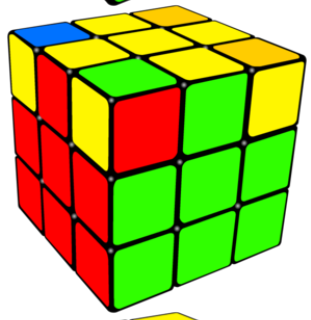
1. Gelbes Kreuz



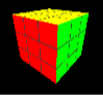
2. Positionierung gelber Kanten



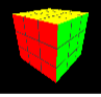
3. Positionierung gelber Ecken



4. Orientierung gelber Ecken



Demo



The End