Ugurtan Can Cetin TIF21B 7925698

# Software Dokumentation
# Rubiks Cube Solver

## Projekt-Beschreibung

Das Projekt “Rubiks Cube Löser” ist in der Lage, einen gültigen Input eines 3x3x3 Zauberwürfels zu lösen. Die Farbkonstellationen, welche gelöst werden sollen, werden mit Hilfe von Farbfeldern in das Programm eingegeben. Daraufhin berechnet das Programm dann einen Weg, den Zauberwürfel zu lösen.
Das Resultat jedoch ist keineswegs optimal. Die Methode, welche genutzt wird, ist die klassische Beginner-Methode “Layer by Layer”.

## Klassenübersicht

### RubiksCubeSolver

enthält die Funktionen, welche zum Lösen des Würfels genutzt werden. Benötigt bei der Initialisierung eines Objektes dieser Klasse ein Cube Dictionary, und eine Instanz der Klasse Movemaker. (Befindet sich in Datei RubiksCube.py)

### UserInterface

Die Klasse die zur Steuerung und Anzeige des Solvers vom User genutzt wird, benötigt bei der Erstellung der Instanz, ein TKinter Objekt, einen Cube Dictionary und eine Instanz von Movemaker. (Befindet sich in Datei RubiksCube.py)

### Movemaker

Diese Klasse bekommt bei Methodenaufrufen den aktuellen Cube als Dictionary mitgegeben und permutiert diesen nach Bedarf. (Befindet sich in Datei mover.py)

### RubiksHelper

Enthält verschiedene Hilfsfunktionen, die nicht direkt zum laufen der App benötigt werden, wie ein Scrambler, ein moveString Parser, der die Strings verkürzt. Es enthält auch Methoden für einen Timer, der jedoch nicht implementiert wurde, jedoch genutzt wird in Zukunft. (Befindet sich in helper.py)

## Modulübersicht

### tkinter

Tkinter ist ein Python Package, welches für UI-Elemente zuständig ist. Es ist ein Toolkit (TK) Interface.
Dieses Package wird genutzt, um ein User Interface für den Rubiks Cube Solver zu erstellen.

**random**
Random ist ein in Python inklusives Modul, welches zur Generierung von zufälligen Zahlen genutzt wird.
Dieses Package wird genutzt um zufällige Zahlen und somit auch zufällige Moves für den Cube zu generieren (siehe: get_scramble(length), Klasse RubiksHelper, Datei:helper.py)

## Resultat

Der Code ist in der Lage, die meisten Rubik's Cube Permutationen wieder zu lösen. Ob wirklich alle gelöst werden können, kann ich nicht direkt testen, da es mehr als 43 Quintillionen Mischungen gibt und ich nicht die Zeit habe, alle zu testen.
Es kann vorkommen, dass bei manchen Rubiks Cube Permutationen es im Code zu einer Endlosschleife kommt, welche dann aufgrund Fehler in manchen Algorithmen für den Würfel befinden. Aber mind. 20 Rubiks Cube Mischungen, welche ich von einer Webseite generiert habe, wurden von meinem Code in unter einer Sekunde gelöst.
Es wurden alle In-Scope Anforderungen, die Input Möglichkeit, eine Misch-Generierung und die Berechnung der Lösung erfüllt. Von den Nice-to-haves wurden leider keine Anforderungen erfüllt, aufgrund der Zeit. Diese werden jedoch in naher Zukunft noch hinein implementiert.

## Mögliche Fehler

Es können Fehler auftreten. Diese Fehler sind dann an einer Endlosschleife zu erkennen. Diese liegen daran, dass der Cube bei bestimmten Fällen in einer bestimmten Art und Weise permutiert wird. Wenn ich mich bei der Entwicklung dieser Algorithmen geirrt habe, werden Endlosschleifen entstehen. Um diese zu beheben, müsste man den Code abbrechen und mit einem Debugger arbeiten. Denn dann folgt ein womöglich sehr langer Prozess an: Schritt für Schritt den Code durchgehen, schauen, dass die Algorithmen, die aufgerufen werden, auch richtig sind. Um dies zu tun, saß ich mit einem Rubiks Cube vor dem Rechner und habe die Algorithmen selber angewendet. Wenn der Cube in meiner Hand dann nicht dem im Code entsprach, wusste ich, dass das der Fehler sein muss.
Außerdem gibt es Würfel-Konstellationen, welche mit illegalen Moves zu erreichen sind (Corner Twists, …). Wenn diese in das Programm eingegeben werden, wird ebenfalls eine Endlosschleife entstehen, weil diese Cubes nicht mit legalen Moves gelöst werden können und das Programm nur legale Moves anwendet.

## Genutzte Hilfsmittel

Für die Implementierung dieses Projektes wurden nur die Vorlesungsfolien für Python und die Dokumentation von TKinter genutzt.

Ugurtan Can Cetin TIF21B 7925698

## Ausblick und Reflexion

Dieses Projekt zu implementieren hat sehr viel Spaß gemacht. Ich bin auch sehr stolz, dass ich in der Lage war, einen Rubik's Cube Löser alleine zu implementieren. Die Logik und Algorithmen habe ich alle selbst implementiert und es wurde keine Hilfe von anderen Programmierern gebraucht. Das Projekt war jedoch keineswegs ein einfaches. Ich habe versucht, einen Analyse Prozess, welchen ich beim Rubiks Cube Lösen anwende, in Computer Code zu übertragen, und habe dies auch teilweise geschafft (oder vollständig falls keine weiteren Fehler beim Testen auftauchen sollten).

Ich werde an diesem Projekt definitiv weiter arbeiten und die Anforderungen/Aufgaben, welche als nice-to-haves definiert wurden, noch implementieren. Besonders freue ich mich über die 3-Dimensionale Darstellung des Würfels. In dem ich auch versuchen werde, den Cube mit der Maus und gut gezielten Zügen zu permutieren. Außerdem arbeite ich an einer Methode, dass der generierte Lösungsalgorithmus auf obsolete Movements gekürzt wird. Die Methode (oder eher der Platzhalter) befindet sich schon in der RubiksHelper Klasse.

Eine Überlegung ist es, das Projekt von einem Python Projekt in ein Unity Projekt umzuwandeln. Bei Unity arbeitet man mit C# und 3D-Modellierung. Aber ich habe mir als Ziel gesetzt, alle Anforderungen erst in Python zu implementieren und dann im Nachhinein das Projekt zu unity migrieren.