**RUBIKS CUBE ENGINE 1.1 DOCUMENTATION**

**Obsah obrázku text, vektorová grafika

Popis byl vytvořen automaticky**User interface

1. 2D NET OF RUBIK’S CUBE – Net shows the current state of the Rubik's cube. An update is performed after each turn
2. RUBIKS CUBE – You can make turns by pressing keys on the keyboard. Every key on the keyboard represents a certain move. For example, key X will perform move X by Rubik's cube official notation
3. CONTROL BUTTONS – CROSS PRACTICE, FIRST LAYER PRACTICE, SECOND LAYER PRACTISE, OLL PRACTICE and PLL PRACTICE start training session for a specific problem in beginner's method. SHUFFLE scrambles cube for practising the whole method. RESET resets the whole program.
4. TIMER – it starts by pressing the first valid move on the Rubik's cube and stops when the cube is solved.
5. INSERT OWN – you can insert your own scramble. By clicking on the colour button, you select a colour to paint the net. When you press CONFIRM button, the program will check the validity of the scramble. If the scramble is correct it will appear on the RUBIKS’S CUBE. If the scramble is invalid, the border colour will turn red.

1. ADVICE FOR OLL AND PLL PRACTICE

* 1. ALGORITHM HELPER – when you hit a part when you must proceed and the algorithm it will appear here. If your move is correct, it will appear there in green colour, otherwise, it will appear in red and you must correct it.
  2. INFO PART – the program will show help messages for the next step
  3. If you want a challenge, you can click on these two and they will turn in black.

SOFTWARE PART

ANNOTATION

The program helps to learn a Rubik's cube by beginner's method. In the beginning, it is difficult to do individual steps and often mistakes are made and it is not possible to practice specific steps. The program solves these problems and helps users to master a beginner's method.

DETAILED SOFTWARE DESCRIPTION

Major files

* rubiks\_cube\_structures.py
  + Contains 4 classes – Rubiks\_cube, Cube, Square\_on\_cube. Rubik’s cube (class Rubiks\_cube) is built by 27 cubes (class Cube) and every cube has 6 sides (class Square on cube) and Rubik’s\_cube\_net

* + Class Rubiks\_cube
    - The main structure of the whole program
    - handles moves inputs
    - generates scrambles
    - calls matrix operation to rotate points in 3D space
  + Class Cube
    - Structure contains 8 major points - vertices of the cube
    - Interface which decides colours for squares
  + Class Square\_on\_cube
    - Stores points of a specific square and its colour
  + Class Rubiks\_cube\_net
    - Stores a net of the Rubik's cube
    - Generate help messages for the user
    - Checks states of Rubik’s cube – solved state, cross…
* utilities.py
  + Contains classes for every utility which can be found in the program
  + Class Timer
    - Class for the timer which is used for measuring time of solving Rubik's cube
  + Class Button
    - Only a rectangle on a monitor which includes a function to detect if the mouse position is inside
  + Class Net\_inserter
    - Users can input their own scramble
    - Decides if the scramble is valid
  + Class Algorithm\_helper
    - Shows user algorithm which can help him to solve a specific problem
    - Has own logic to decide which algorithm should be shown
  + Class Info\_window
    - The rectangle which can be shown to the user monitor to display a simple message
  + Class Parts\_solved
    - Stores times of specific steps and show them to users
* matrix.py
  + Includes function for matrix multiplying and matrices for rotations
* algs.py
  + The file includes algorithms that are later used for shuffling the cube and in certain situations, they are shown for users
* settings.py
  + Separate file for storing settings to store them in one location for easier use
* rubiks\_cube\_engine.py
  + The main file of the program, information in other files are combined here to run the application

LIBRARIES

* PYGAME
  + To display shapes on a display
  + Handles keyboard and mouse inputs
* DATETIME
  + It is used for the implementation of a timer
* TYPING
  + Used for better understanding of variable types and return values of functions

WORK PROGRESS

1. Function for matrix multiplying.
2. Classes Rubiks\_cube, Cube, Square\_on\_Cube (basic structure)
3. User inputs – keyboard presses
4. Rubik’s cube net
5. Implementation of a scramble inserter – I had a problem with that, I had to rewrite a function that decides the colours of squares. It was a good idea because it saved me a lot of hours later. The function now uses class Rubik’s cube net as an input – it is easy to change colour now.
6. Timer
7. Algorithm helper – I wrote a lot of functions in class Rubik’s cube, which information I can use to show helpful messages.

INPUTS REPRESENTATION

* MOUSE
  + It is used for inputting own scramble
* KEYBOARD
  + Users can control Rubik’s cube movement as it is described in the previous part

Invalid inputs are not possible - invalid keyboard presses are ignored.

UNFINISHED PARTS

* Users should be able to control the cube by click and mouse movement
* Help for cross and first two layers – User should be able to select a piece and the program should give him advice for this specific piece
* Automatic solver
* Own scramble checker is not perfect and it is not possible to decide the validity of a scramble without solving a cube – but it catches about 90% of wrong scrambles

FEELINGS

* I think I did a decent job. I had experience with programming in the past, but my largest projects had about 600 lines of code, so this was quite a challenge. I think the code is divided into files and classes as it should be. It is easy to modify.
* Before I started writing the code, I should have written down the structure of the program, because later I came across some annoyances and some parts could have been written better

TEST DATA

* Unittests for functions are written inside files where certain function is located
* User interface tests was done by me and other person just by using it