Design Documentation

Rubik's Cube AI

Zhaowei Gu

Overview

Take the opportunity of an AI class project to observe different possibilities to solve rubik's cube. I have started with research on how to develop an AI that will solve rubik's cube without any knowledge, but in the end most of the paper states that this method only works if you have shuffled 1-7 times.

I have also tried to develop my own AI without any knowledge using BFS, theoretically it should find the possible result, but in the end, it take much longer to find a result that work, and sometime it won't be able to find a result

Then I decided to try to develop an AI with human knowledge and found some bugs in the rubiks cube itself. I have splitted one job into 7 different steps just like how a beginner would learn Rubik's cube. In the end it turned out to be great. It will be able to solve any Cube I throw at it and solve it very fast.

Context

I have been playing Rubik's Cube for years, most of times human would just follow an algorithm to complete the cube, so the idea of finding a way to solve the cube without any algorithm comes into my mind, maybe computer could help us to find a way to solve Rubik's cube in a much more less steps. So here I am to explore.

Goals and Non-Goals

- Study and learn different way to build AI for Rubik's Cube
- Try to build non human knowledge involved AI
- Try to build a algorithm AI

Milestones

- Read different way to solve rubik's cube without human knowledge
- Understand that is the minimum number of moved needed to finish a cube
  https://tomas.rokicki.com/rubik20.pdf

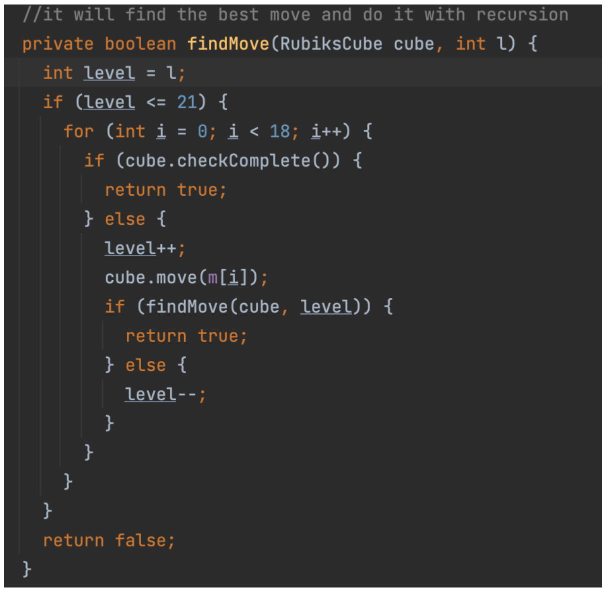
- Every standard 3x3x3 Rubik's cube can be solved in a maximum of 20 moves from any starting position, no matter how scrambled. In 2010, programmers found that 20 is the colourful puzzle's so-called "God's number", a name they selected to suggest that even a deity couldn't solve a Rubik's cube any faster.
- Watch youtube to see how other people write Rubik's Cube
- Start writing code for non human knowledge involved AI
- Start writing code for build a algorithm AI

Existing Solution

- Algorithm solve
- Search backward from completed cube

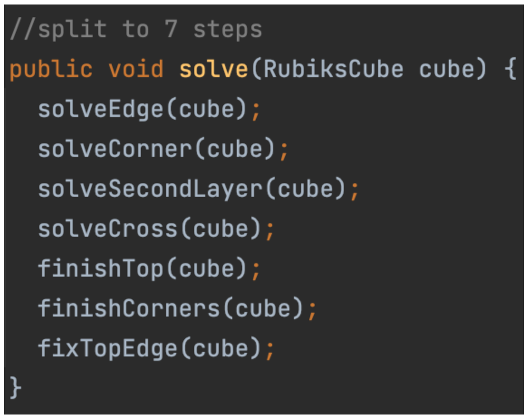
Proposed Solution

1. Uses BFS to look for the best way to solve the cube, from a non complete cube to complete cube. It will try every possible combination, but it's going to take a long time, and sometime it will not find a result, I do not like this solution

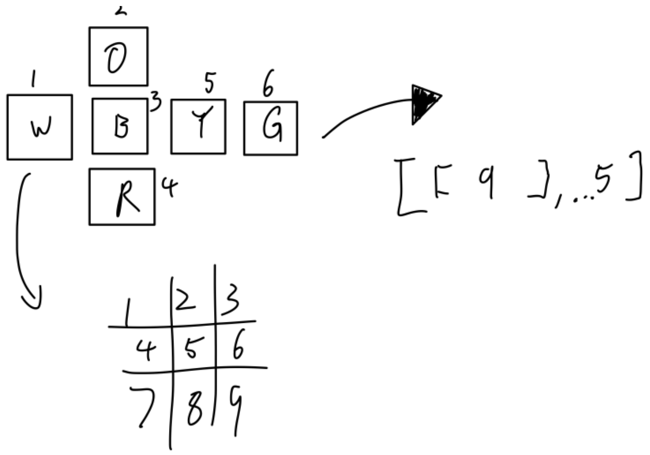


```
//it will find the best move and do it with recursion
private boolean findMove(RubiksCube cube, int l) {
  int level = l;
  if (level <= 21) {
    for (int i = 0; i < 18; i++) {
      if (cube.checkComplete()) {
        return true;
      } else {
        level++;
        cube.move(m[i]);
        if (findMove(cube, level)) {
          return true;
        } else {
          level--;
        }
      }
    }
  }
  return false;
}
```

2. Uses algorithm to solve the cube

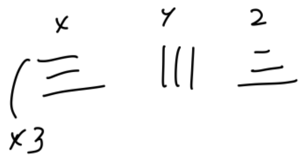


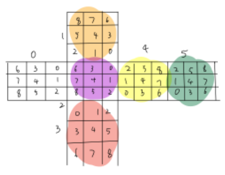
```
//split to 7 steps
public void solve(RubiksCube cube) {
  solveEdge(cube);
  solveCorner(cube);
  solveSecondLayer(cube);
  solveCross(cube);
  finishTop(cube);
  finishCorners(cube);
  fixTopEdge(cube);
}
```



PeasyCam to Center Cube.

Use matrix transformation to turn the Cube

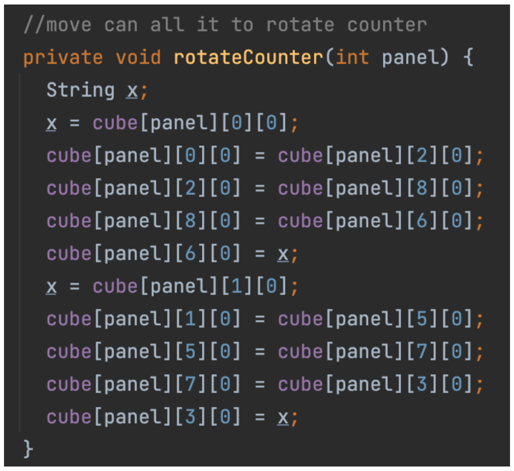




1×3 ×3 ×3
9 27 81

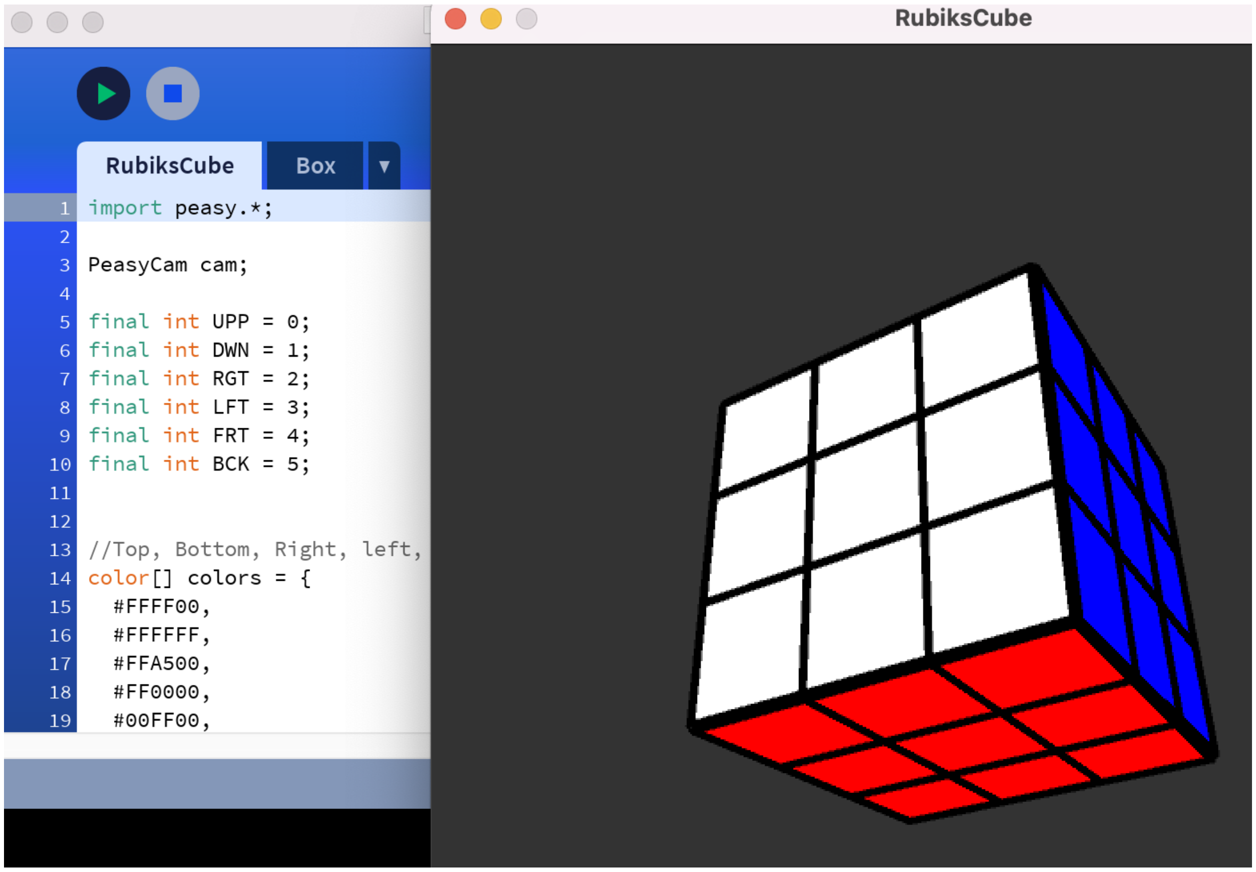
Uses 3D array to represent, Rubik's cube each contain 0-8 spot
move along the x y z axis each axis has three moves and each move can move to two directions in total there are 18 moves.

- Builded the fundamental logic of how cube should move with very ugly code that nobody want to read



```
//move can all it to rotate counter
private void rotateCounter(int panel) {
  String x;
  x = cube[panel][0][0];
  cube[panel][0][0] = cube[panel][2][0];
  cube[panel][2][0] = cube[panel][8][0];
  cube[panel][8][0] = cube[panel][6][0];
  cube[panel][6][0] = x;
  x = cube[panel][1][0];
  cube[panel][1][0] = cube[panel][5][0];
  cube[panel][5][0] = cube[panel][7][0];
  cube[panel][7][0] = cube[panel][3][0];
  cube[panel][3][0] = x;
}
```

3. Use processing try to build a visilized cube



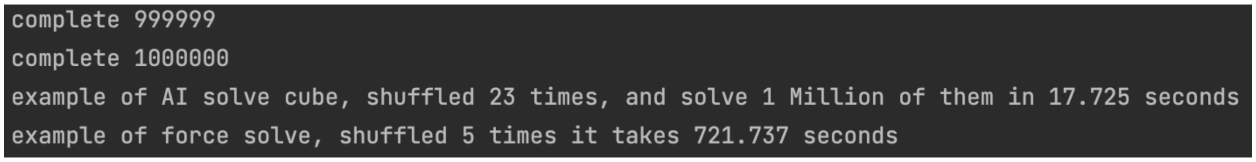
```
import peasy.*;

PeasyCam cam;

final int UPP = 0;
final int DWN = 1;
final int RGT = 2;
final int LFT = 3;
final int FRT = 4;
final int BCK = 5;


//Top, Bottom, Right, left,
color[] colors = {
  #FFFF00,
  #FFFFFF,
  #FFA500,
  #FF0000,
  #00FF00,
```

Alternative Solutions

Next time I should try another way to do the Force AI to make it smarter, but it will use much more resources such as memory and storage

I also think that there are possibilities that I could improve the Ai solver logic to make it solve faster and less repeat processes

Testability, Monitoring and Alerting



```
complete 999999
complete 1000000
example of AI solve cube, shuffled 23 times, and solve 1 Million of them in 17.725 seconds
example of force solve, shuffled 5 times it takes 721.737 seconds
```