John Flynn

Description of Data Structure

* For this assignment I used a three-dimensional array of cubie objects which makes up my cube. I also added another Cover object to sit on top of each cubie’s sides in order to get different numbers to appear. In order to keep track of the movement of the individual cubie’s I used matrix operations within a two-dimensional array by rotating my half of Pi and then rounding the answers to ensure there is not weird trig errors. To keep the Covers attached to each cube as they rotate, I attached a normal vector to each cube and maintained their location by using this rotation matrix equation in relation to each normal vector:A picture containing object

  Description automatically generated
* The GUI is fully 3D ready and you can use your courser to navigate around it and adjust the viewing angle. I acknowledge that I was not able to get numbers on the covers of the cube in time, however I did get an idea last night that I was unable to fully implement due to time constraints. In an effort to prove that the cube is actually functioning with rotations and a randomizer I added colors temporarily.

Running the data structure

* To run the program, you can open the files in the Processing program and click run in the top left corner. You can also run it in terminal. If you are on Windows or Linux, use the processing-java program that's in the download. If you are on Mac OS X, there's an option in the Tools menu within the Processing application to install the command line tool. The command is:
  + processing-java --sketch=/full/path/to/your/sketch/dir --output=/path/to/output/dir --force –run
  + This link was very helpful for me - <https://github.com/processing/processing/wiki/Command-Line>

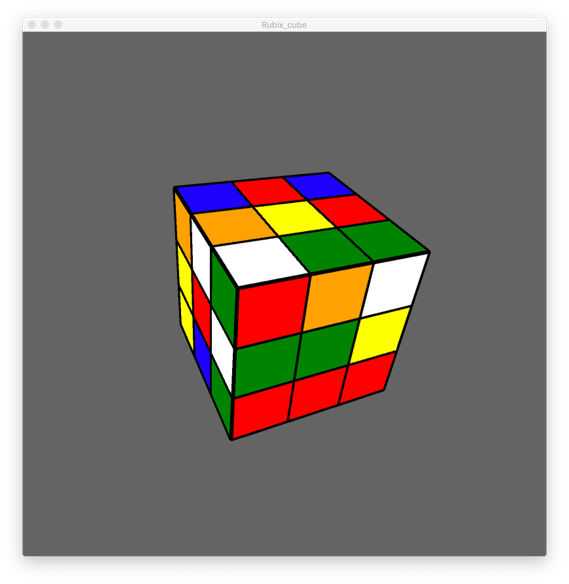
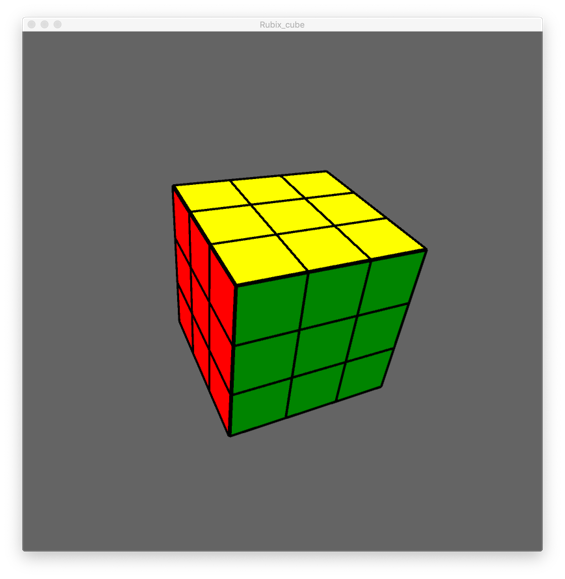
Description and Running the Randomizer

* The randomizer works by pressing the space bar. That creates a random double between 1 and 50 which in then passed into a loop. Once the loop beings running it generates a separate random integer between zero and five. Those six integers represent the different possible moves that will be executed as the loop executes. The number of rotations that the cube went through is printed out. I had trouble with inputting with Processing, so I had to take an alternate route of randomly generating the number of moves to randomize the cube.
* To run the randomizer the cube must be running and simply hit the space bar when the cube is selected. You can also perform the individual moves by using the 1-6 keys.

My Heuristic

* As I thought through this problem, I first thought that I would attempt to work out a Max Manhattan distance. This was done by calculating the maximum distance of each of the corners from their original orientation and taking their sum. Then calculating the maximum distance of each of edge to its original orientation and summing those. From there, simply take the maximum of both those sums, and we have a heuristic. As I went to test this Idea, I began to discover that while the logic seemed solid it had some problems that ended up making it non-heuristic. It came to me that when I make a move of the cube, I do not move one single cubie like I was originally considering but rather eight. This means that calculating and summing each of the edge and corner cubes is actually overestimating the number of moves by a factor of eight. Therefore, in order to make my original heuristic admissible I had to do my original theory and divide the final value by eight.
* I believe it is admissible because each cubie has an original starting place therefore Manhattan distances are plausible. In addition, I believe that my addition to divide by eight makes the heuristic admissible by making the Manhattan distance summing model the way the actual cube rotates.

Example of my GUI Output

* 

What I Learned and Problems along the way

* In this project I had a great time going back through my matrix algebra and physics coursework in an effort to accurately represent rotations with the cubies and their respective covers. I also learned Processing which has great upside when it comes to designing and implementation GUI’s. I did have growing pains and I did come up short a in a few areas. I believe that I found a way to finally ad numbers to the covers rather than colors late last night, however I ran out of time and I plan on continuing to attempt to implement that element before the next project. The other issue I plan on continuing to tackle is the randomizer. While the one I have implemented works correctly, I had a very hard time getting Processing to take an input. It works just fine with individual keys, however after that things get weird. I even attempted to store individual key presses into an array of chars to no avail. Overall it was a great and challenging experience that allowed me to challenge myself and shake the coding rust off of my java-based programming.

Credit

* I got inspiration and guidance designing the GUI from these YouTube Videos-
* <https://www.youtube.com/watch?v=9PGfL4t-uqE>
* <https://www.youtube.com/watch?v=EGmVulED_4M&t=2356s>