CS463G Program 1: Representing the Pyraminx

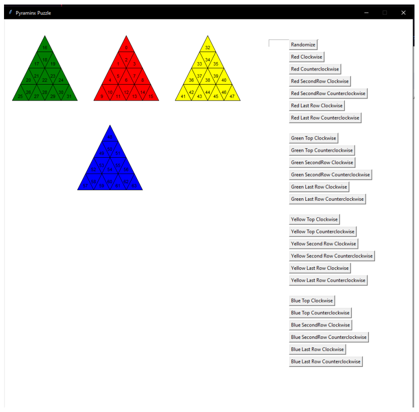
Implementor's Notes
Louis Lin
Dr. Judy Goldsmith, CS463G
09/11/2024

ABSTRACT

This project involves the design and implementation of a 4x4x4 Pyraminx puzzle model which comes with a randomizer and basic GUI for visualization. The main objective is to create the data structures and rotation functions that allow the Pyraminx to move in its legal moves. The randomizer will allow for the scrambling using legal moves by specifying rotations on a given face and direction. In addition, the project comes with an admissible heuristic that approximates the minimum number of moves required to return the puzzle to its solved state, without overestimating this value.

GENERAL APPROACH

This assignment required the implementation of data structures for the Pyraminx puzzle. The puzzle state is stored in a Pyraminx class using a dictionary called faces, which represent the 4 sides, with each face (A, B, C, D) being represented by a 4x4 2D list of colors (R, G, Y, B). This structure allows for manipulation of each individual triangle on each face.


Pyraminx Puzzle
Randomize
Red Clockwise
Red Counterclockwise
Red SecondRow Clockwise
Red SecondRow Counterclockwise
Red Last Row Clockwise
Red Last Row Counterclockwise
Green Top Clockwise
Green Top Counterclockwise
Green SecondRow Clockwise
Green SecondRow Counterclockwise
Green Last Row Clockwise
Green Last Row Counterclockwise
Yellow Top Clockwise
Yellow Top Counterclockwise
Yellow Second Row Clockwise
Yellow Second Row Counterclockwise
Yellow Last Row Clockwise
Yellow Last Row Counterclockwise
Blue Top Clockwise
Blue Top Counterclockwise
Blue SecondRow Clockwise
Blue SecondRow Counterclockwise
Blue Last Row Clockwise
Blue Last Row Counterclockwise

Next is the representation of the GUI. Tikinter's **Canvas** widget is used to draw the puzzle onto an app. Buttons are created and stored in a list of buttons which are called through the Pyraminx app. The GUI uses a list called triangles which stores information about each triangle, represented by a dictionary containing properties: {id, row, col, type, x, y, color}.

Now for the hardest part of the assignment: Rotations. A Master Pyraminx puzzle has much more complex rotation mechanics than a normal Pyraminx. So each face rotation in a certain direction (clockwise or counterclockwise) is defined as a function. Each rotation manipulates the colors of specific triangles with their unique IDs and reconstructs the color data which is used for randomization and eventually solving algorithms.

Randomization took the least amount of time due to its reliance on the rotation functions that I created prior. The user inputs a random number and clicks Randomize. The randomize function would pick out of the 24 functions to run, over and over again until the length of user input.

THE HEURISTIC

The heuristic of the Pyraminx is the approximation number of legal steps to get to the solved state. This project looks for the admissible heuristic, which means this number is guaranteed to not be greater than the actual number of steps – a lower bound. My breakdown of how to calculate the admissible heuristic is through the use of misplaced edge pieces.

Edge pieces are a piece with an adjacent sticker. There are 12 edge pieces meaning that the maximum number of heuristic would be 12 steps, which is admissible because it would most likely take much more steps to actually solve the Pyraminx. In order for the heuristic to be 12, all edge pieces are either misplaced or misoriented.

For implementation, we would need to identify the edge pieces, check each edge piece for misplacement, then calculate the heuristic based on misplacement.

This method is admissible because each misplaced edge will require at least one move to correct it, but never fewer than one. Even though it might take multiple moves to fully correct an edge piece, this heuristic only increments 1, ensuring it does not overestimate.

WHO-DID-WHAT

This project was quite difficult for me to visualize myself so I had help from some classmates in how to visualize this in my head. I created a large majority of the concept of the code myself and used Chat-GPT for helping me code the 24 rotate functions at a faster pace, since honestly, if I manually did it myself, it would have easily taken over 5+ hours.

CONCLUSION

From this assignment, I learned somewhat how a Pyraminx operates, even though it's still cloudy. I also learned how to use Tkinter to create GUI shapes and lines as well as buttons that can be used. In regards to AI, it really focuses on the algorithm and deeper understanding needed to fully understand how obscure rotations work, since humans can't really understand manually how to rotate pieces back to their original state. The importance of algorithms and otherwise the heuristic serve as guides and estimators.