CS463G - Modeling the Gearball

Programming Buddies - Samyak Piya & Aastha Bhatt

-A description in English of your data structure

We made separate classes to represent each individual piece, "piece.cpp", each of the 6 sides (top, bottom, left, right, front, and rear), "side.cpp" and the gearball itself, "gearball.cpp".

A Piece object has a color (R,G,B,P,O,Y) and position (GL.GR.GT.GB.C.CTL.CTR.CBL.CBR.ETL.ETR.EBL.EBR)attribute and methods to manipulate these values. The positions GL stands for Gear-Left, C -> Center, CTL -> Corner-Top-Left, ETL -> Edge-Top-Left and so on.

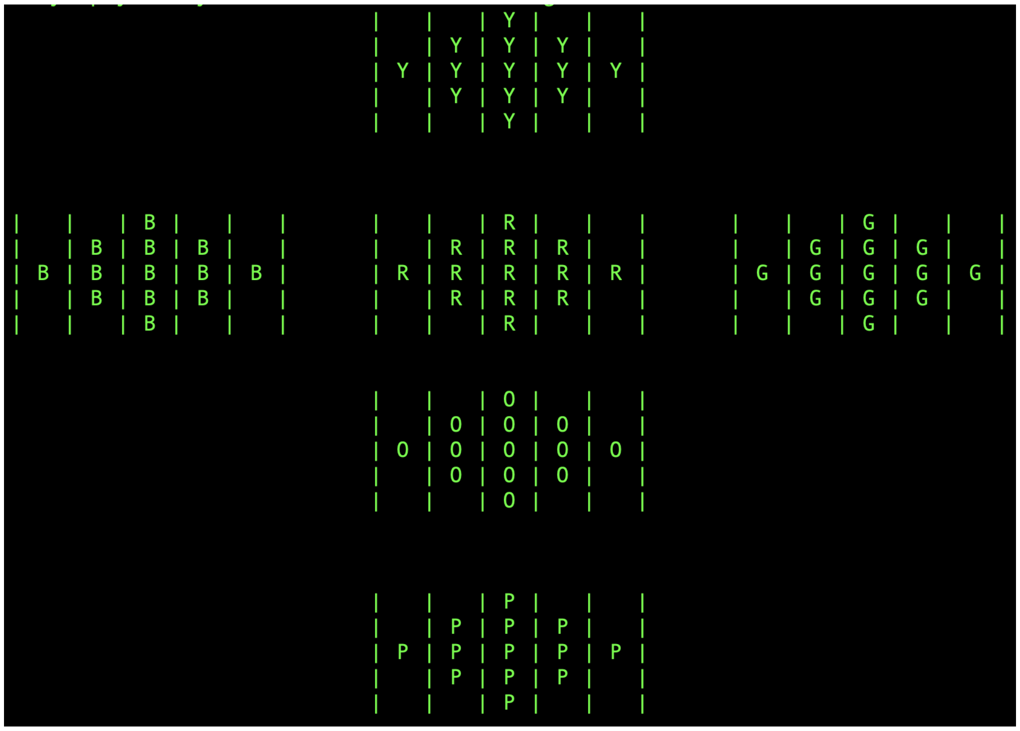
A Side object has a face (top, bottom, left, right, front, rear) and a 5x5 2D Array of Piece objects. The different pieces (center, corners, edges, gears) are stored in the 2D array as follows...

	0	1	2	3	4
0			P		
1		P	P	P	
2	P	P	P	P	P
3		P	P	P	
4			P		

...for each of the 6 different colors (the above being Purple).

Finally, a Gearball object has three Side object arrays named sides, previousSides, and solvedSides, all of length 6 (total number of sides). It also has 8 different rotation functions to perform top, bottom, right, and left rotations in both clockwise and counterclockwise directions.

A gearball object starts with a solved state as such:



```
                           |   |   | Y |   |   |
                           |   | Y | Y | Y |   |
                           | Y | Y | Y | Y | Y |
                           |   | Y | Y | Y |   |
                           |   |   | Y |   |   |


|   |   | B |   |   |      |   |   | R |   |   |      |   |   | G |   |   |
|   | B | B | B |   |      |   | R | R | R |   |      |   | G | G | G |   |
| B | B | B | B | B |      | R | R | R | R | R |      | G | G | G | G | G |
|   | B | B | B |   |      |   | R | R | R |   |      |   | G | G | G |   |
|   |   | B |   |   |      |   |   | R |   |   |      |   |   | G |   |   |

                           |   |   | O |   |   |
                           |   | O | O | O |   |
                           | O | O | O | O | O |
                           |   | O | O | O |   |
                           |   |   | O |   |   |


                           |   |   | P |   |   |
                           |   | P | P | P |   |
                           | P | P | P | P | P |
                           |   | P | P | P |   |
                           |   |   | P |   |   |
```

The randomize() method in a gearball object can be used to change the state of a gearball to an unsolved state.

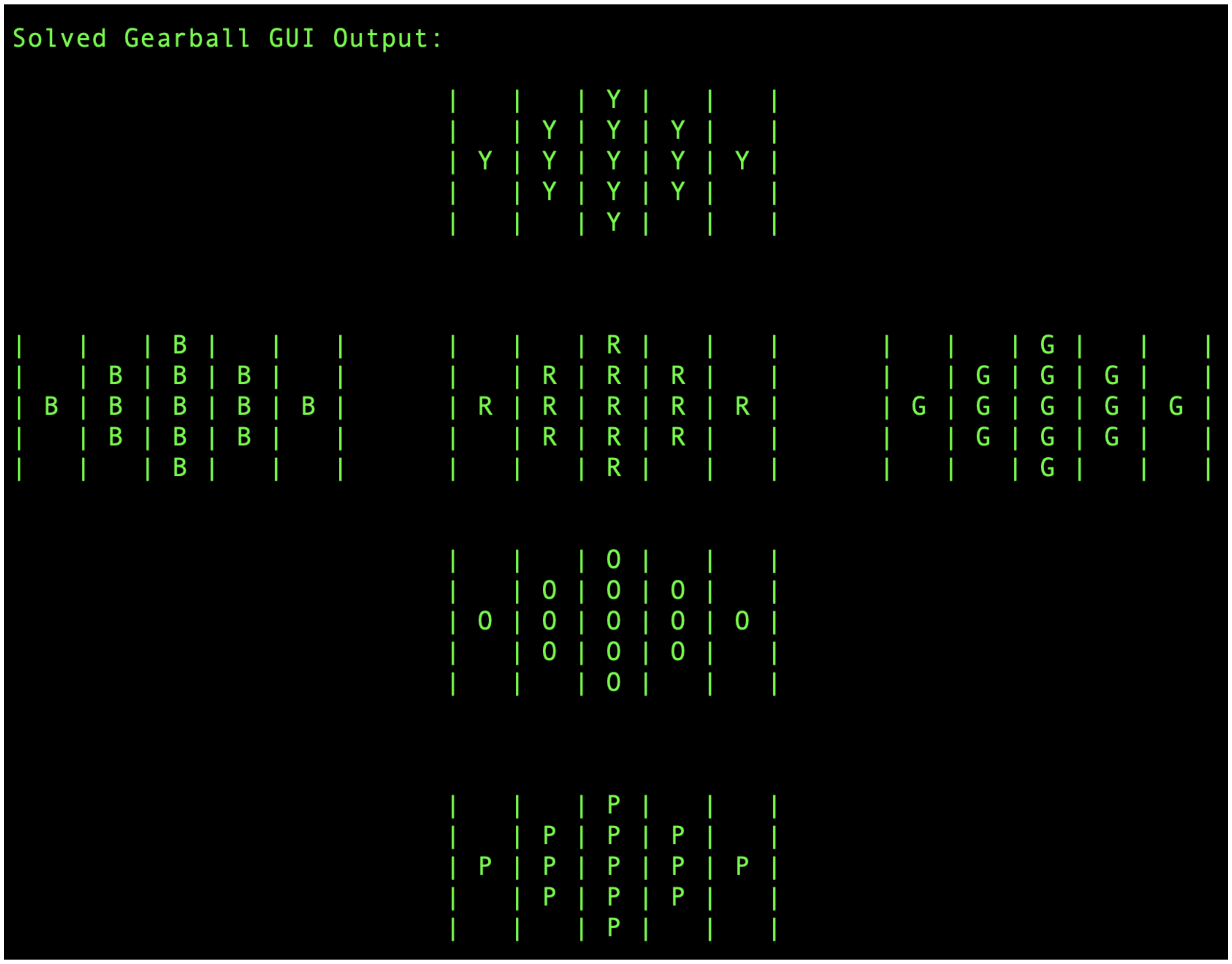
-Code for the data structures, along with instructions on how to run it

The code for the data structures can be found in “piece.cpp”, “side.cpp”, and “gearball.cpp”.

To run the code, unzip “Gearball.zip” and open up a terminal. Change the directory in the terminal to where the Gearball folder is located. Use the following command in the terminal to compile and then run the following code:

```
$ g++ main.cpp -o main
$ ./main
```

-An example of the GUI output



```
Solved Gearball GUI Output:

                           |   |   | Y |   |   |
                           |   | Y | Y | Y |   |
                           | Y | Y | Y | Y | Y |
                           |   | Y | Y | Y |   |
                           |   |   | Y |   |   |


|   |   | B |   |   |      |   |   | R |   |   |      |   |   | G |   |   |
|   | B | B | B |   |      |   | R | R | R |   |      |   | G | G | G |   |
| B | B | B | B | B |      | R | R | R | R | R |      | G | G | G | G | G |
|   | B | B | B |   |      |   | R | R | R |   |      |   | G | G | G |   |
|   |   | B |   |   |      |   |   | R |   |   |      |   |   | G |   |   |


                           |   |   | O |   |   |
                           |   | O | O | O |   |
                           | O | O | O | O | O |
                           |   | O | O | O |   |
                           |   |   | O |   |   |


                           |   |   | P |   |   |
                           |   | P | P | P |   |
                           | P | P | P | P | P |
                           |   | P | P | P |   |
                           |   |   | P |   |   |
```

-A description of the randomizer

The randomizer function uses the <stdlib.h> and <time.h> C++ libraries to generate a pseudo-random integer in the range of 0 to 7. This creates 8 possible random int generations which are enough as our program consists of exactly 8 different rotation functions.

A switch statement is used to perform corresponding rotations for each number selected from the 0-7 range. A helper function, restartRandomization() is used to avoid negation effects when a clockwise rotation is followed by a counterclockwise reaction or vice-versa (for example, top clockwise rotation followed by a top counterclockwise rotation).

-Code for the randomizer, along with instructions on how to run it;

The code for the randomizer and its helper function restartRandomization() can be found in "gearball.cpp" after unzipping "Gearball.zip".

To run the code, unzip "Gearball.zip" and open up a terminal. Change the directory in the terminal to where the Gearball folder is located. Use the following command in the terminal to compile and then run the code:

```
$ g++ main.cpp -o main
$ ./main
```

-Your heuristic, clearly described and justified, including an argument that it is admissible;

The heuristic we came up with is calculated by dividing the number of out-of-place tiles by the total number of rotations that could have caused those tiles to be misplaced.

If we rotate the left and right sides of the gearball, we displace the left and right layers at the front, rear, top, and bottom sides. Including all the faces we rotated and all the gears, this results in the total number of moves to be 52.

We are arguing that the heuristic is admissible as determining the number of tiles that are out of place is an underestimate and it is also normalized by dividing the value by the maximum number of tiles that can be displaced in a single rotation.

-A statement of what you learned from this assignment.

We learned better strategies for modeling a relatively complex real-world object. Breaking down a problem into smaller chunks, and making detailed mockups/designs of the problem makes both programming and thinking about ways of data representation for the problem much easier (e.g. making separate classes for Piece, Side, and Gearball). We also learned how to work better in a team setting, coordinate ideas, and delegate tasks. We also realized that working on a problem for consecutive hours straight can be detrimental to your backbone.

-The who-did-what for you and your programming buddy.

We both worked collaboratively on the design of the GUI, coding the Gearball, heuristics, and writing descriptions together. But here is a more specific who-did-what:

Aastha (Ash):

- Came up with and implemented the idea to break down the Gearball puzzle into smaller chunks and create separate classes for pieces, sides, and the gearball itself.
- Sketched the mockups of the different states of the gearball for each unique gearball rotation.
- Coded getter and setter functions, constructor, and putIn() function in piece.cpp.
- Coded getter and setter functions and printing functions in side.cpp.
- Coded rotate_rcw(), rotate_lcw(), rotate_rccw(), rotate_lccw() functions, and all printing functions in gearball.cpp.
- Coded rotaeFace90Deg() and rotateFace180Deg() functions in gearball.cpp.
- Recommended use of srand() instead of rand() to come up with pseudo-random integers that give unique rotation sequence every time the program is run.
- Implemented the randomizer helper function, restartRandomization(), to remove code redundancy.

Samyak (Sam):

- Came up with the idea and implemented 2D Arrays to represent each piece on a side of a gearball with its associated color and have a “ ” character in unused array spaces.
- Defined constants that avoided the use of naked values in code making the program less prone to errors.
- Coded isValidColor() and paint() methods in piece.cpp.
- Coded initialize(), generateFace(), and the constructor in side.cpp.
- Coded rotate_tcw(), rotate_bcw(), rotate_tccw(), rotate_bccw() functions in gearball.cpp.

- Coded saveSolvedState(), solve(), generateSides(), and makeCopy() functions in gearball.cpp.
- Used rand() and coded the randomizer function to generate a value in the range 0 to 7 to correspond to each of 8 unique gearball rotations.
- Fixed minor miscalculations in rotation functions that wrongly represented the rear face in the GUI.