Week 08

23/4 – 24/4

Interaction:

1. Show the board, cars, trucks. Also, use a different colour to indicate the target car
2. The exit should always on the 3rd line of our 6*6 board
3. The 3rd line should only contain one car, which is our target car
4. Show how many moves the player made
5. Show the minimum moves required to solve this board
6. The user can choose levels, reset, go back to the main menu
7. Give hints to the player if they need
8. Start page, winning page …
9. Other functionalities

25/4 – 26/4

Code:

1. Generate the cars/trucks situation by keeping their coordinate
2. Must generate a board that solvable, i.e. only one car in the 3rd line
3. Car and board interaction
4. Find the shortest path to solve the board after every move that the user made
5. Use a 2D array to represent the board – 0 is space, 1 is car/truck, 2 is the target car
6. JavaFX to make the animation

27/4 – 28/4

Summary:

1. Divide-and-conquer
2. I am working on the algorithm part, which finds the best moves
3. Next week, find the algorithm, which can be BFS or Astar search

Week 09

30/4 – 1/5

Think about the algorithm

1. Go through their way of representing the vehicle and the board
2. I think the problem can be solved by considering the vehicle as the vertex, and the space between the vehicles as the edge. Moreover, use breadth-first search to find the shortest path
3. Update the graph after each move
4. The problem with this algorithm is ‘huge complexity.'

2/5 – 3/5

Think about the algorithm

1. An appropriate way to implement the algorithm of finding the minimum move is using BFS.
2. Using HashMap<> to keep the path <key = previous vehicles’ position, value = current vehicles’ position>
3. Using Stack<> to keep the finial path after the breadth-first-search by tracking back step 2
4. Using Queue<> to do breadth-first-search

5/5 – 6/5

Coding:

1. Finished the first version of ShortestPath.java
2. BFS the possible movement of every vehicle
3. Use the HashMap to record the path
4. Push every possible move to the queue
5. Moreover, push the shortest path to the stack
6. The stack contains the best path
7. Return the number of minimum moves and the position of vehicles after each movement
8. Uploaded to Github

Week 10

7/5 – 8/5

1. After finish the algorithm of finding the shortest path to solve the problem, I tested it and found there are some mistakes. The hashMap<> is not as I thought, the key cannot be found, and keep finding what's wrong with my code. Since when the element changes, the hashCode() changes and the ArrayList<> cannot be equal
2. Fix the hashCode() problem, solve the grid, and get the quickest way to solve the board

9/5

1. Review the algorithm code and other parts

11/5 – 12/5

Merely the algorithm of finding the minimum moves to solve the board. Moreover, make sure it generates the correct answer by testing a couple of grid which found online. Considered the queue expanded nodes would not be too much difference, I do not think to change the BFS to Astar is a good idea. Moreover, I also discussed with teammates to decide what need to do next.

The UI design will add the extended functions like different levels of games, give players hints, and show them how many steps they already made.

13/5 – 14/5

I thought the level of games would depend on the minimum moves they have to make to solve the board. However, after discussing, it turns out that limit the maximum moves of users made is more reasonable.

I also thought about how to give users hints. What the existing apps contain is after the players click the hint button, the board is reset and instruct the users to play from the beginning. The advantage of it is the backend doesn't need to generate/calculate again and again. And also show the users that it does exist the minimum moves as we said.

15/5 – 16/5

However, the details on how to instruct them finish the games is hard. Before I talked to my teammate who in charge of the frontend, I thought it's clear to highlight the vehicle that need to move and show an arrow in front of it. However, it would bring the frontend team be too much work. So the other solution I came up with is highlight the vehicle which about to move and shows the right place to put the vehicle by another color highlighting.

17/5 – 18/5

Separated works again. Chuyi and I are assigned to combine the frontend hint button and my ‘minimum move’ method.

Week11

21/5 – 22/5

Since my computer died, I worked with Chuyi. We added ‘hint’ feature which is when the user clicks the ‘hint', the vehicle that needs to move is highlighted. Also, also shows that how many moves the user currently made and how many moves you need to make to win.

23/5 – 24/5

We fixed some bugs such as hint button and steps counter. Also, drew the UML diagram.