**Final Project Written Report**

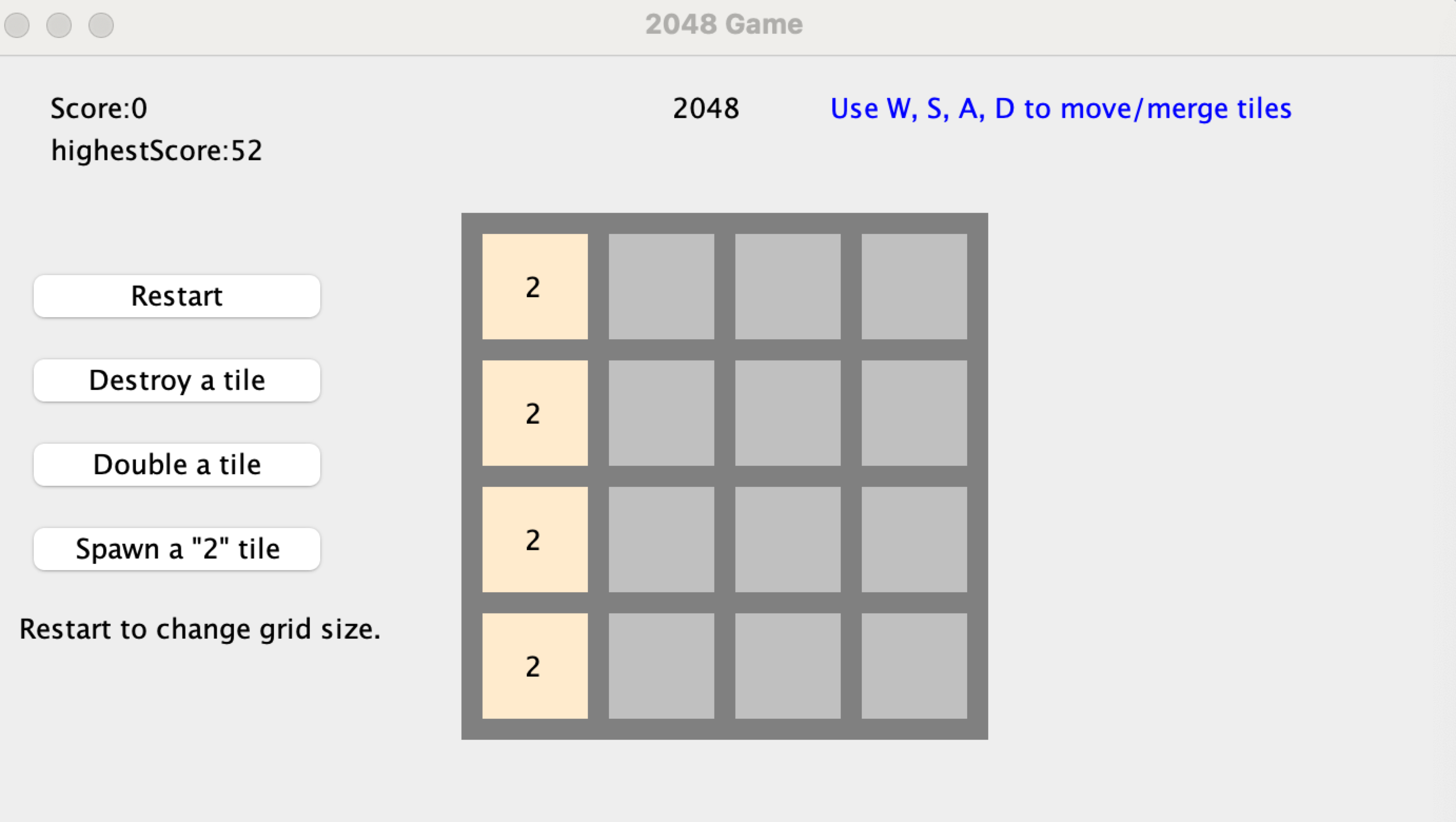
Project 2048 Game

CS240

Ting Gao, Cancan Huang, Jacqueline Tan, Yushi Yao (alphabetically order of last name)

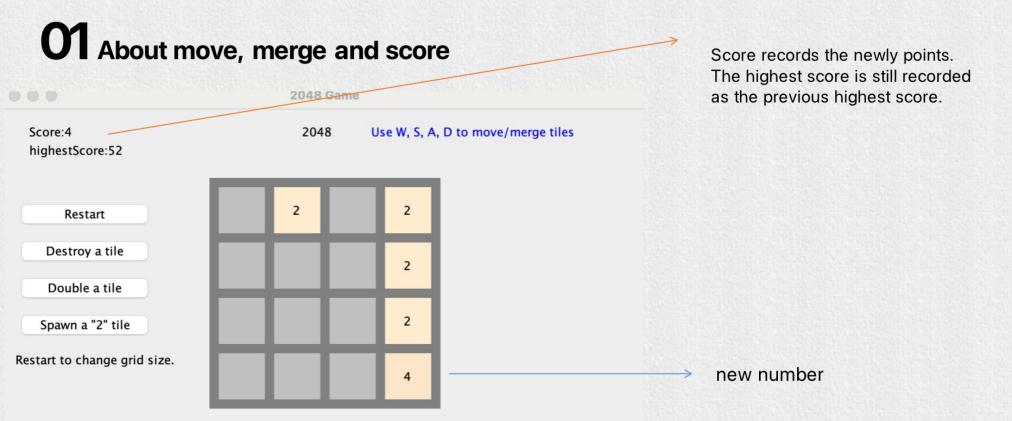
GitHub Repo: <https://github.com/D4KLRE/Project_Game_2048>

**Introduction:**



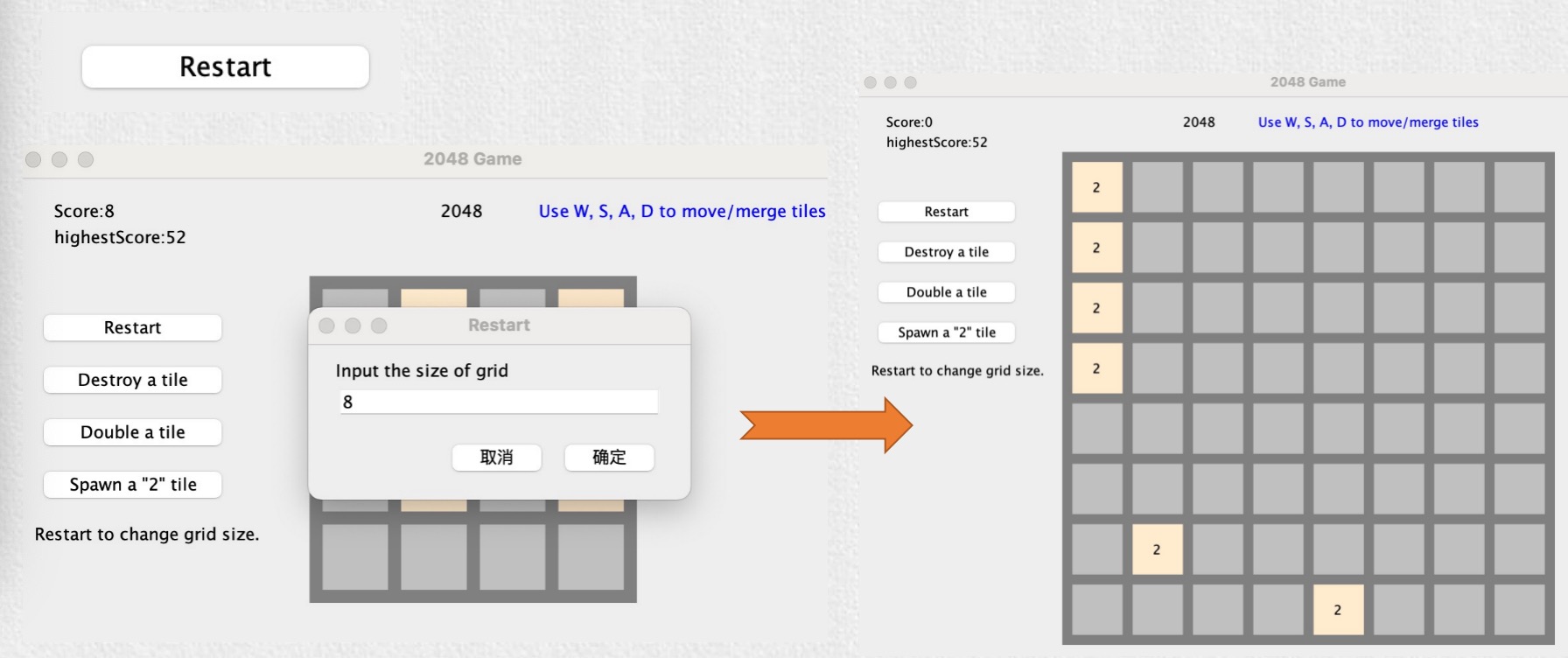
You need to control all the cubes to move in the same direction, two identical numbers collide and merge to form their sum. After each operation, a random 2 or 4 will be generated, and finally a "2048" block is considered victory.

**Demo:**

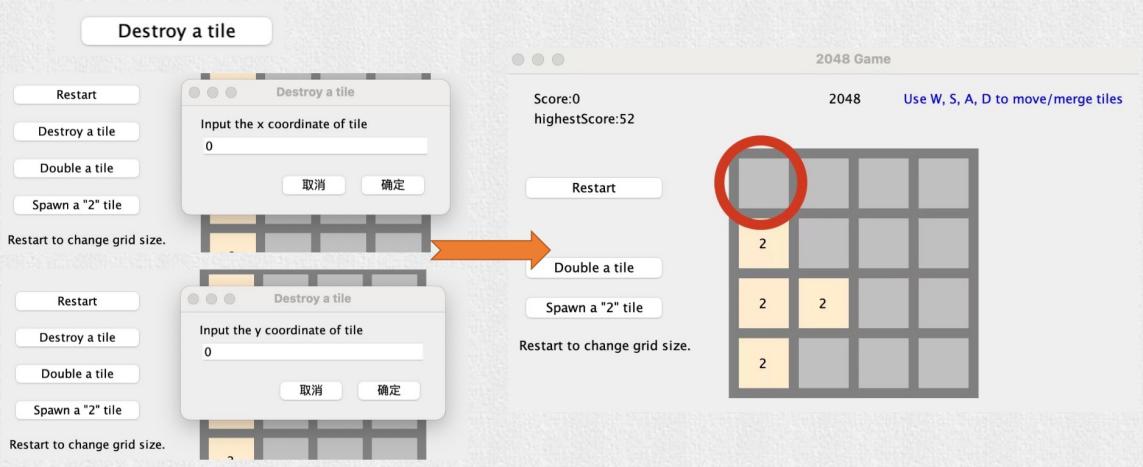
The first is about move, merge and score. This step uses the D on the keyboard to move one step to the right, adding the two identical twos to form a new number four. Score records the newly synthesized 4 points. The highest score is still recorded as the previous highest score.

About Function Keys

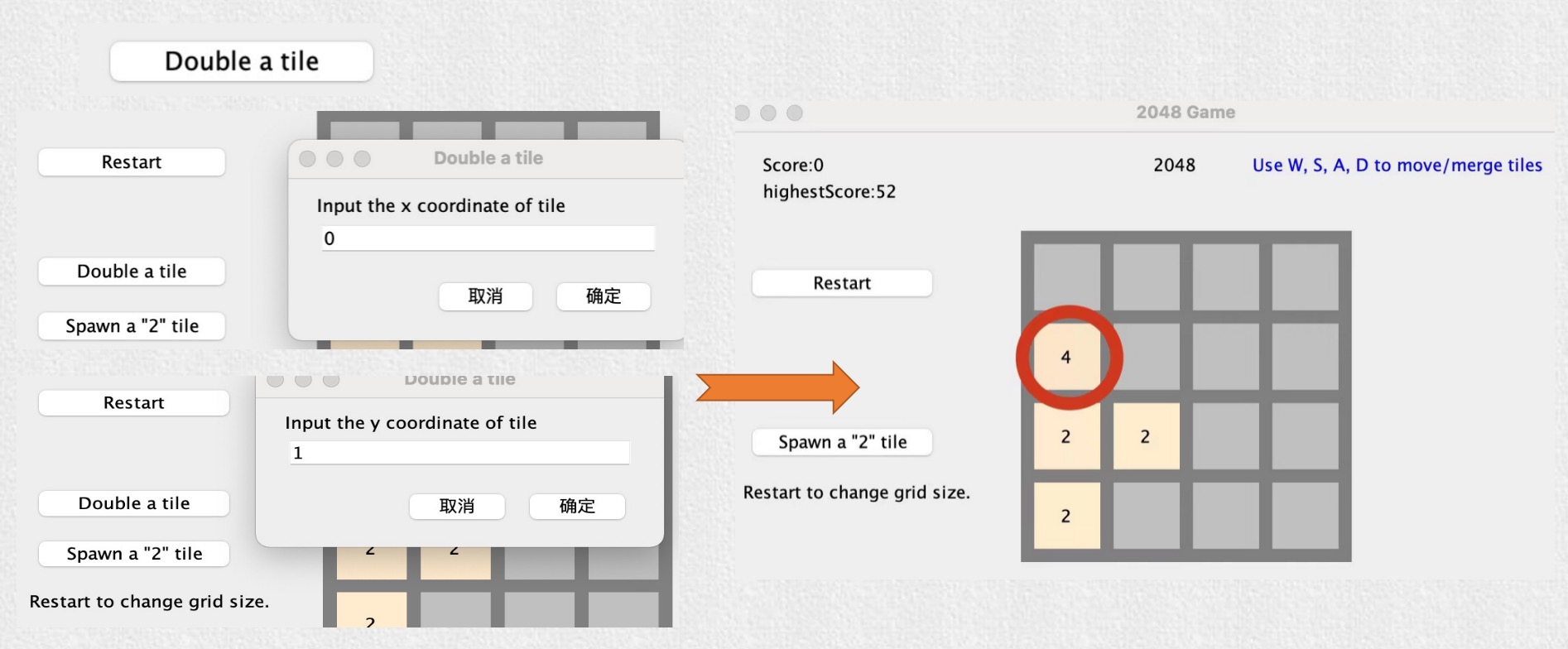
For restart, we can change the size of the grid.



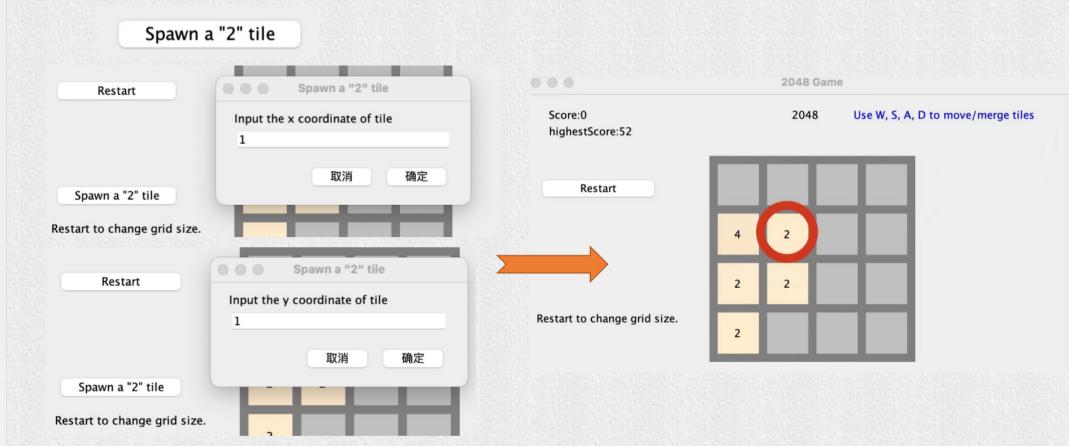
Destroy a tile allows us to specify coordinates to delete numbers.



Double a tile specifies coordinates that make the number twice as large.



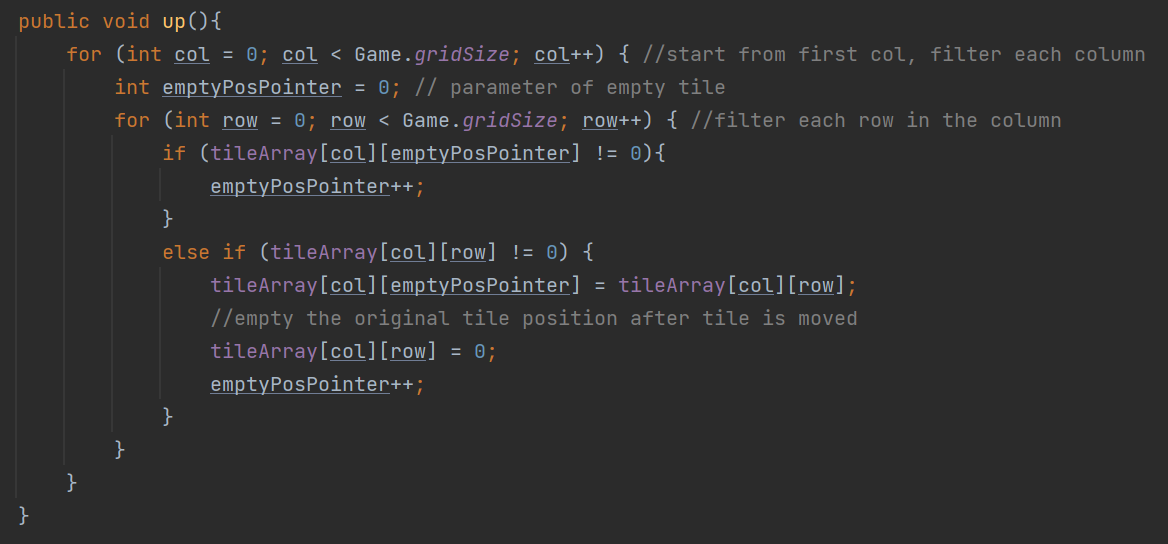
Spawn a 2 tile generates 2 in the specified coordinates.



These buttons add interest to the game and can only be used once.

**Design:**

This is the move function where if you click the WASD keys, the blocks should move to left, right, up, and down.



For example, the idea of this up function is to move all the tiles in a column to go farthest up the border/tile and repeat the process for the remaining columns. To do this, a for loop is called to achieve each idea as shown in this code snippet, creating a nested for loop. The first for loop is to move all the tiles upwards in a single column, while the next for loop is to traverse between the other columns. Because of this, each move function has a time complexity of O(N^2).

While making this function, other alternatives like making a single function for all directions or a function for a horizontal/vertical movement was considered. However, the best solution that came up was a separate method for each movement.

The background color of tiles differs depends on their value. While testing the game, we found that identical background color made the grid difficult to read. My eyes got tired after I played for a while. Hence, we decide to assign different color to each value. In addition, higher value will have a deeper background color, making the game more intuitive.

The move method and merge method are separated, so merge method needs to test whether the closing tiles have the same number or not. Also, the method needs to ensure the correct position if two tiles have the same number after each movement.

merge

To make a distinction between 2048 games found online, we made the size of game alterable. To match up with this change, more than 1 “2” tile can be spawned each round depends on the size of grid. But game could be more difficult with these changes. Hence, we added several single-use abilities (destroy a tile, double a tile, undo, etc.) to ease the difficulty.

**Implementation:**

Two-dimensional array is used to represent the grid of 2048 game. We made this decision because we believe two-dimensional array is the most similar and intuitive one to a n \* n grid and 2048 game requires mass data modification.

At first, we stored positions of all tiles in an array and pick a random one among them to spawn a new “2” tile. However, we soon found that if a majority of tiles are already nonempty, running time increased since a nonempty position could be randomly chosen more than once. In addition, if all tiles are not empty, the game will fall into an infinite loop trying to find an empty position. Therefore, we implemented our own IntArrayList to optimize. The size is variable and we can remove nonempty positions from the list so they won’t be picked again.

We achieved the undo method by implementing our own stack of saves, SaveStack and SaveNode. Every time the user performs an action (move and merge up/down/left/right), a save of grid will be generated and pushed into the stack. Instead of recording every change of tile to achieve the undo method, this is more efficient and easier.

SaveStack, SaveNode and IntArrayList are all specially implemented for this project to reduce space complexity, we didn’t write additional methods or we remove them if they were never called in the code.

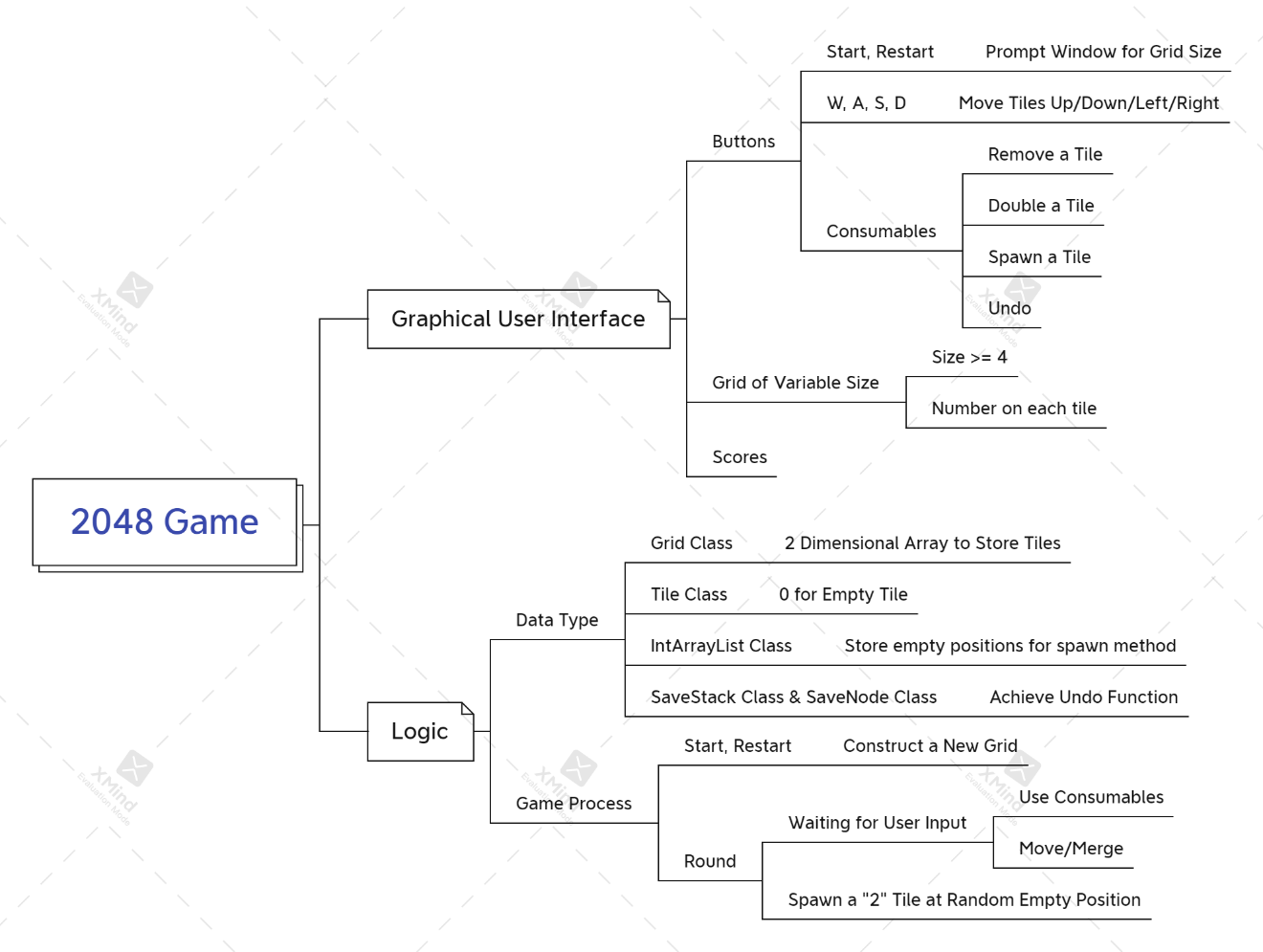
**Testing:**

Each time we made a modification, we would write test code to search for bugs, dash out methods not written yet, and add breakpoint or System.out.println to make sure code is working properly. For example, when we haven’t finished spawn method to spawn “2” in random empty position, we added “2” manually to test other code/method.

Beyond that, we tested many times by playing the game, creating/modifying/removing highest score file, inputting invalid number after being prompted to choose the size of new grid or position of tile to be destroyed, doubled, or spawned.

We have fixed multiple bugs after testing. So far, no more bugs were found.

**Goals:**



This is the mind map we made at the beginning. Undo ability, IntArrayList, SaveStack and SaveNode Class was later added to improve the game. We have met all goals above by cooperation. And we didn’t go in a different direction because we found no serious problem while working on the project.

**Future Work:**

As this is a private group project, this game has not been played by the general public. We could let this project be tested by other users to debug other features in the game. We could also finalize the game's GUI by adding sound effects and creating a title, menu, or game over screen.

We have created some unique functions where we could spawn a tile, delete a tile, double a tile and undo a movement. We could make the game more unique by adding some arcade modes such as timed mode or frenzy mode, where it will give tiles different power-ups that spawn at random or when a certain value/score is reached.

**Contributions:**

**Ting Gao**

Making mind map, framework of codes, GUI, buttons method, keyPressed method, spawn method, implementing IntArrayList, SaveStack and SaveNode, and writing check-ins.

**Cancan Huang**

Writing buttons methods, implementing score and highest score function, making framework of presentation slides

**Jacqueline Tan**

Writing core logics, move, of the game, including up, down, left, and right method

**Yushi Yao**

Writing merge methods, including up, down, left, right, designing different colors for different values of tile, implementing IntArrayList, and editing videos

**Final Thoughts:**

**Ting Gao**

This is the first time I take part in a CS group project. It was definitely not plain sailing but both interesting and valuable for me. I will surely keep this in my memory.

**Cancan Huang**

The game is now perfect, although every detail in the process took some time. On the whole, it went well.

**Jacqueline Tan**

This game has been tested by members of this group and has run smoothly, so overall this project could be marked as a success for all group members.

**Yushi Yao**

I consider the experience of this group project very enjoyable. We communicate with each other through Telegram and share coding files through GitHub. Ting signed us each with a clear goal and we all sticking to the plan.