StackPointers’ final project -- LianLianCan

Inspiration:
This project is an adoption of a Japanese board game Shisen-sho which is a game for matching and cancelling tiles with a board of Mahjong tiles. Our implementation of game is based on the rules of the original game with better user experience.

In our project, we implemented the original design of the game, and we also added some more contents such that users have more freedom to choose from with their game experience. We allow the user to choose the themes for the images of the tiles. Shuffle and hint buttons are provided if the user is stuck during the game. We also provide a exciting background music. A time bar is shown at the bottom of the program let the user know how much remaining time they have.

Background:
One member of our team, Haipu, used to spend a lot of time on this game when he was a child. Hence, we are inspired to improve this game with different theme of pictures such as rage manga and photos of Bucknell professors to let Haipu revisit his childhood memory and other Bucknell student experience what Haipu’s childhood was like.

Introduction:
Our game consists of a board with different tiles. The objective of the game is to eliminate tiles by selecting tiles with the same images before the time is up. However, there is one rule that needs to be followed in order to connect the tiles: the player can only connect two tiles with up to three line segments passing through empty spaces.

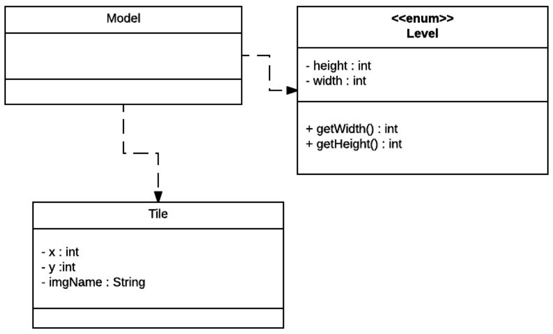
We need to find out if two tiles are able to match through a backend object called Model. The Model object contains a two dimensional array list of Tile objects, which are essentially the abstraction of the picture tiles on the actual game board. Cancelled tiles are represented as null. From the frontend Controller, we are able to find out which two Tiles the user clicked on and parse them into the Model. Each Tile object contains the x, y position of the tile within the game board and a path for image. To find if there’s a path, first check if there’s a vertical or horizontal way between two positions. If there’s not, try to see if they can be connected with one turn. One turn is basically a horizontal line with a vertical line, and the turning point needs to be null as well. Also, this point is should be on a horizontal or vertical line from one Tile. Similarly, we can check for situation with two turns, which can be seen as one horizontal or vertical line plus a one-turn situation. Starting point, turning points and ending point are all recorded which are used to show the path.

In order to show the path, since we used tile pane for the game board, it’s difficult to add other objects like lines on the tile pane, so we decided to change the opacity to show the path. The path’s opacity is then updated to the normal one on the next thread cycle.

For the shuffle method, all tiles are stored in a temporary array and shuffled, updated in the Model and then repainted to the game board.

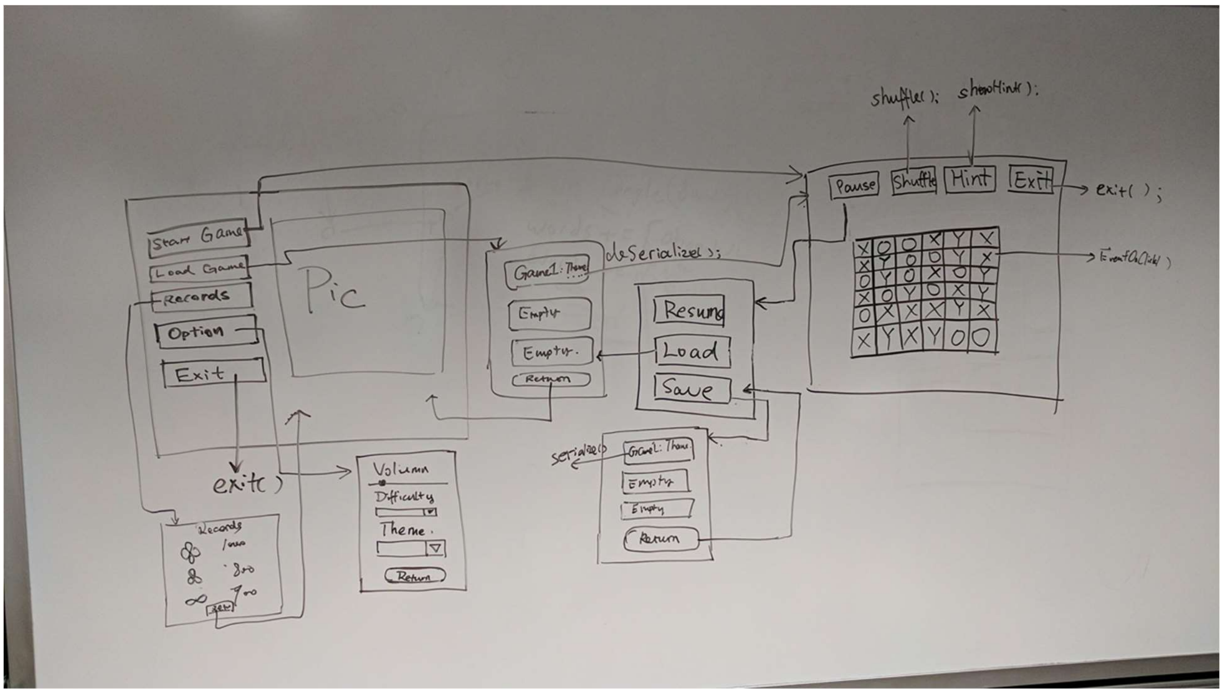
Whenever levels up, the Model object will be updated with new parameters.

To store and load the game, the Model is made to be serializable.

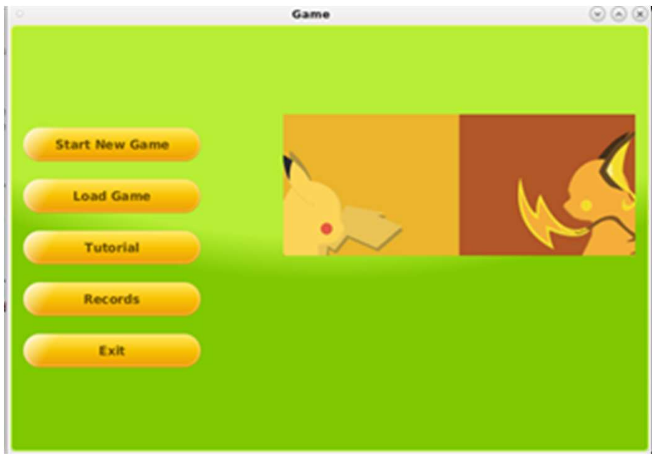


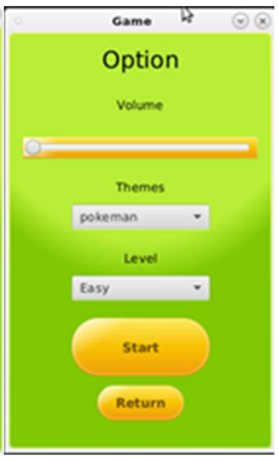
Instructions on using the program:

We plan to use a self-explanatory user interface so that the user can choose to start the game, load the game, choose difficulties. When the user load a game or start a new game, the window will be switched to a new window that includes the main playground of the game. The user will select the tiles they want to connect with the mouse click, and the playground will respond to their choices by eliminating the game.



Later on during the implementation we did some minor adjustment to our user interface and decide to make it prettier by using the CSS.

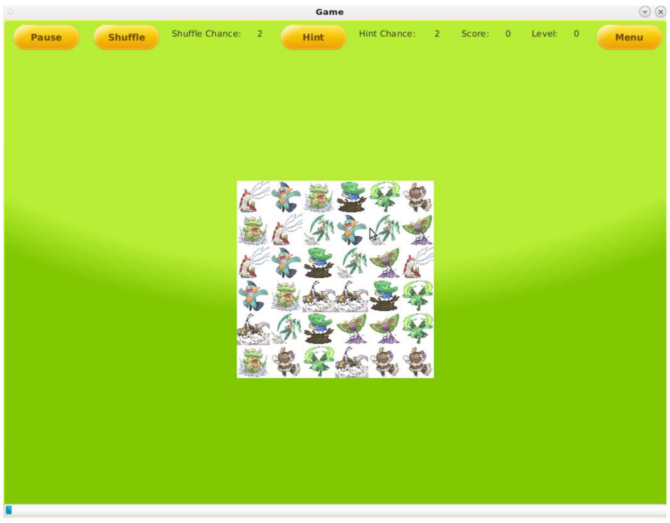




Volume

Themes pokeman

Level Easy



Shuffle Chance: 2

Hint Chance: 2

Score: 0

Level: 0