CSCI205 Final Project

Design Manual

StackPointers ---- Xing Fu, Zilin Ma, Haipu Sun, Junjie Jiang

Introduction -- general description of the system

This projects is a cooperative work of a Scrum team of four. The scrum members worked together on a board game that is defined by the front end implemented by JavaFX and CSS, and a back end design that is produced using Object Oriented Design (OOD).

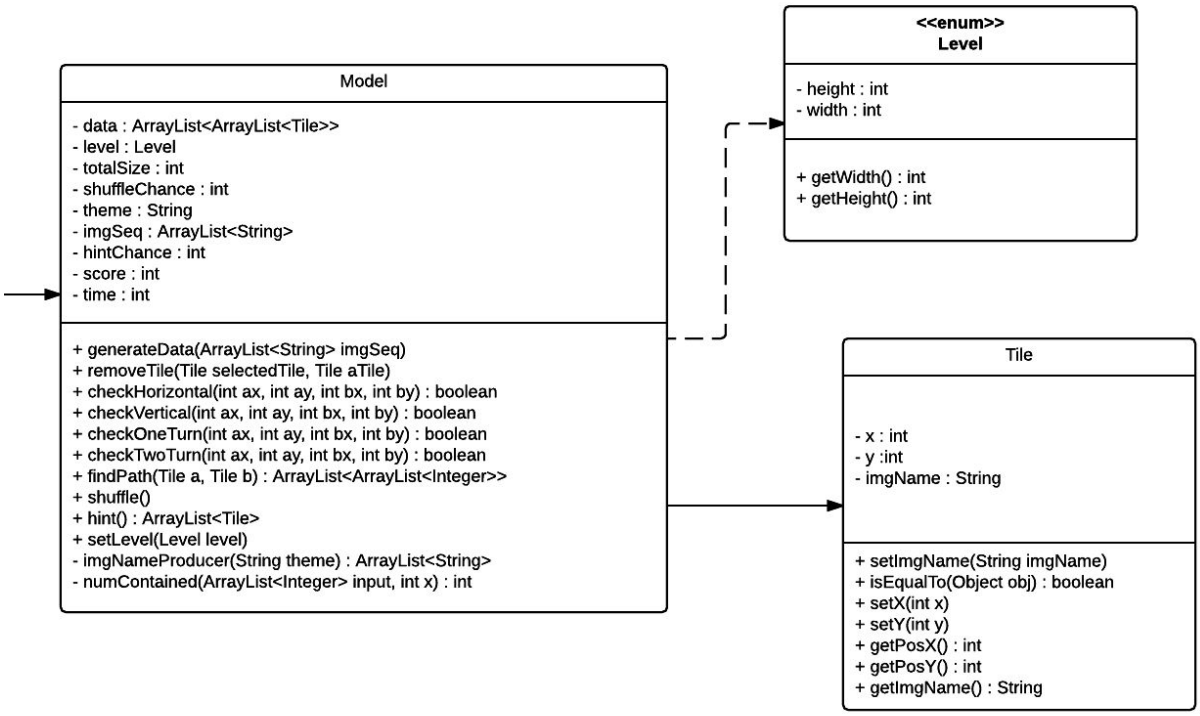
List of User Stories that are implemented

The table below includes the user stories that were implemented:

As a user, I want a clock to show the remaining time.
As a user, I want a menu provided for me to choose to save game
As a user, I want a menu provided for me to choose to resume the game.
As a user, I want a menu provided for me to choose to load game
As a user, I want a menu provided for me to choose to exit the game.
As a user, I want a pause button, so that we can pause the game and open a menu.
As a user, I want a pause menu that allows me to save game, resume game, or load game.
As a user, I want to be able to navigate between the menus using simple buttons.
As a user, I want to be able to change themes for each game.
As a user, I want a scoring system so that I can compare the score with my friends.
As a user, I want hint items, so that I can have a motivation to play, when I am stuck.
As a user, I want shuffle items, so that I can have a motivation to play.
As a user, I want the game displays the line between the pictures I match to have a direct view of the game process.
As a user, I want to start the game while I click start game button.
As a user, I want the UI to be beautiful so that I could love the game more.
As a programmer, I want to dis-inherit the Tile Class so that we can serialize the Model Class.
As a user, I want a Game class so that I can separate the UI and the actual game.
As a user, I want to play the music while the music is playing.
As a user, I want to be able to load the game.
As a user, I want to be able to save the game.

As a programmer, I want to fix load the game button.
As a user, I want to ascend to the next level if I eliminated all the pictures before the time runs out.
As a user, I want to lose if I do not eliminate the pictures before the time runs out.

OOD Design



We used three classes for the back end: the Model class, the Level enumeration class and the Tile class. These classes have related functionalities, with the Model class depending on the Level class and directly associated with the Tile class. This structure provides a clear and neat hierarchy of classes.

Model	
Knows the level Knows the theme Knows the number of hint chance Knows the number of shuffle chance Knows the score Knows the time Generates and keeps the tile data Finds the path of two tiles Checks for game over	Level Tile

Our Model class serves as the main class for the game, controlling the tile pane, remaining time, scores, level, theme, hint chance and shuffle chance. Most importantly, the Model class runs our major path finding algorithm, which is the core algorithm of our game. Since the Model class controls the tile pane and the Tile class controls the tiles, the Model class depends on the Tile class and tiles are stored in the Model class using the arraylist data structure.

Level	
Knows the height of the tilePane Knows the width of the tilePane Upgrades level	

Our Level class allows users to choose difficulties. Since it is an enum class, it allows potential update of the difficulty level of the game in future versions. It also facilitates the construction of board by providing width and height.

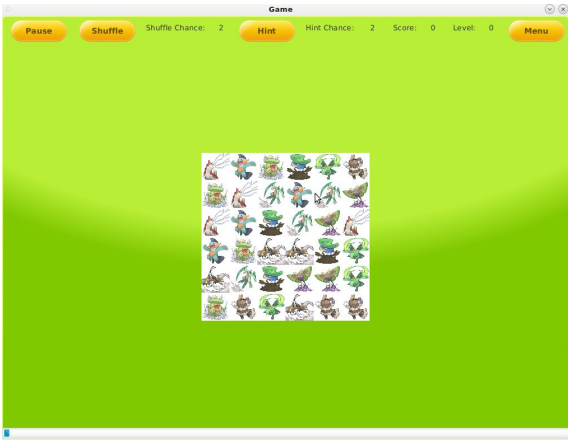
Tile	
Knows its own position in the tile pane Knows its image name	

Our Tile class is served as the most basic element on the back end, which stores the relative position and image path for each rectangle.

Controller Classes

The front end GUI takes care of the interactive part. We have eight controllers, each corresponding to a different scene in our game. We can switch between scenes by clicking the buttons in each scene.

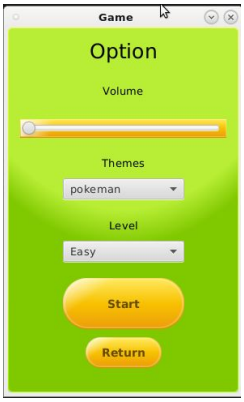
Our game scene controller is the most important controller in our program. This controller is responsible for initializing a model, starting a game, ending a game, recording hint chance, shuffle chance and scores, showing remaining time, playing music and exit. It also provides a pause button, which once clicked will pause the game and pop out the pause menu.



Our tutorial controller shares similar configuration with the game scene controller , providing an auto-hint functionality that can guide the first time player how this game works.

Our menu controller is served as the controller for the initial interface, including start new game, load game, records and exit. These buttons will lead to different scenes and have different functionalities.

Our option controller presents many options of game for players. Volume slider can control the background music, themes can choose themes of tiles and level can choose the difficulty of the game. After users confirm their choices, they can click on start button to start the game or return button to go back to menu.

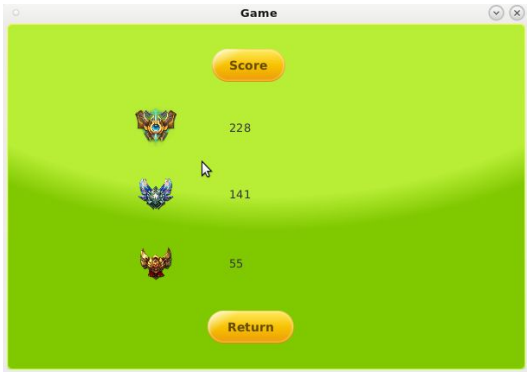


Our pause controller will present three buttons, which are resume, save game and load game. Once players arrive pause window, all tiles in tilepane won't be visible to player and the time bar won't keep counting until resume button is clicked. Save game button pops out save game scene and load game button pops out load game scene.

Our save controller will present three slots for players to choose to store their game progress and a return button to go back to previous window.

Our load controller will present three slots stored with the game progress saved by players previously and a return button to go back to previous window.
Note: save game menu and load game menu have same appearances.

Our record controller is to present the top three scores in game history.



Utility

The Save/LoadUtility is a helper function that serialize or deserialize the Model object into a .ser file such that the GameSceneController will be able to read the Model object from a .ser file and draw the tiles onto the TilePane.

AudioUtility

Our audio utility will load a music file and play it when player are enjoying our game. At first, we planned to use Mediaplayer in Javafx to help us play the music file, but afterwards we found that mediaplayer is not supported by our netbeans, so we decided to use clip.