Emoji Crusher

Jacob Malcy, Kyle Stewart, Vincent Liu
Project 6 - CSCI 4/5448 OOAD

FInal State of System Statement

Features Implemented:

i. Scoreboard
ii. Emoji Selection
iii. Game
    1. Timer
    2. Emoji Match System
    3. Score system

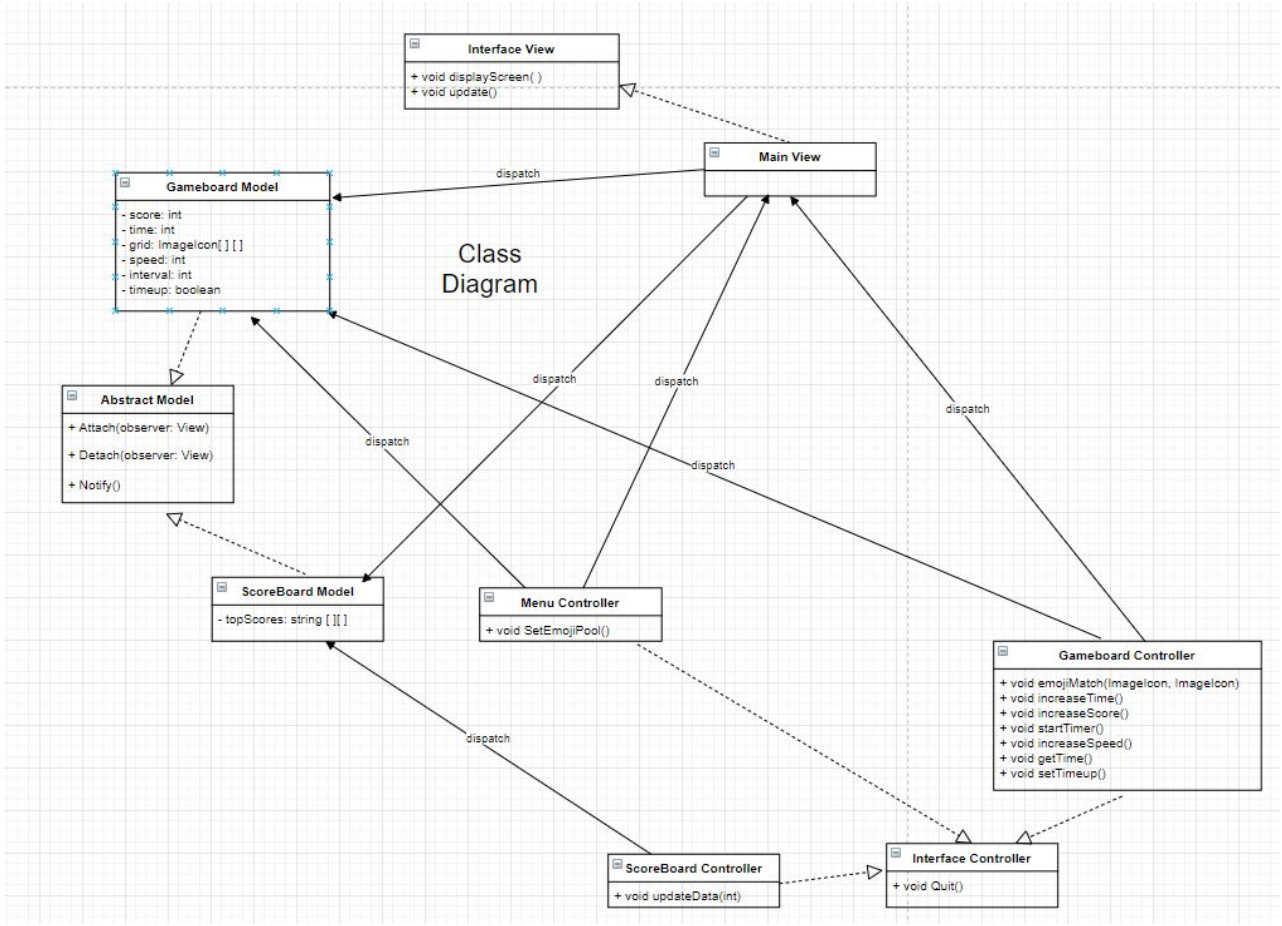
We implemented the score system by sending and receiving scores to and from a SQLite database. In addition, we stored our emoji selections inside the same database in a different table. When the game is relaunched after the first play, the game loads in your emoji selections from the previous game state. Our matching system allows the player to pick two adjacent emojis and swap them to try and get a “3 or more in a row” which is a match. When a player gets a match, the time and score increase. When the time runs out, the game ends and prompts the player for their username to log the score in the database. The timer ticks down as the game is played. The rate at which the time decreases will increase the longer the game runs. The menu has 4 options: Play game, select your emojis, view the scoreboard, or quit. Originally we considered making gravity for the emojis to fall down into replace matched emojis, but decided against it for time constraint reasons. When emoji’s are matched, the above elements still shift down but it’s index based instead of gravity.

Final Class Diagram and Comparison Statement

We ended up completely restructuring the project, as seen between the two diagrams below. The first diagram is from project 4, while the second is our final class diagram. In our original diagram, we did not plan around connecting the game to the database for emoji loading, emoji saving, and saving highscores. We completely changed how we used controllers. Instead of having multiple controllers for different interfaces or views, we now have one single controller that controls several views. We discovered that swing provides a threading system for the timer so we ended up not making an observer/observable relationship pattern manually.

We try to mimic a MVC design pattern but we're not sure if we did it right

Project 4 Class Diagram



Project 6 Class Diagram (FINAL)

Third-Party code vs. Original code statement

b. A clear statement of what code in the project is original vs. what code you used from other sources - whether tools, frameworks, tutorials, or examples - this must be present even if you used NO third-party code - include the sources (URLS) for your third-party elements
   i.

Statement on the OOAD process for your overall Semester Project

c. List three key design process elements or issues (positive or negative) that your team experienced in analysis and design of the OO semester project
   i. Positive: good work environment, good collaboration & communication
   ii. Positive: We learned new technologies, strategies, and design patterns
   iii. Positive: Learning how to **COPE** about not knowing everything

Video Outline / Script:

1. Introduce all team members
2. Discuss generally who was responsible for what
    a. Jacob “MVP”:
        i. Game board logic
    b. Vincent: load data from and to the database, game over submission, scoreboard
    c. Kyle: emoji selection board logic, formatting submissions, game over submission
3. Demonstrate final application, identify the technologies used and primary functions
4. Reflect on anything that did not go as planned or that you would do differently
    a. Jacob
        i. No auto-matching system
        ii. Shipping with libraries bad
        iii. UI sizing is aaaaaallllll messed up
        iv. Fighting the IntelliJ UI Designer
    b. Kyle
        i. UI didn’t work seamlessly between different operating systems
        ii. Borked limits on the emoji select
        iii. Score on game interface screen does not update properly
        iv. Git Kraken broke multiple times
    c. Vinny
        i. Git LFS is a nightmare
            1. Initial commit of emojis included extras and clogged up VCS
        ii. Game interface class is enormous and should be shot
        iii. Didn’t plan around an empty database initially