Java Game - Team 15
David Nyberg, Ryan Cooper, Ryan Davis, Charlie Davies

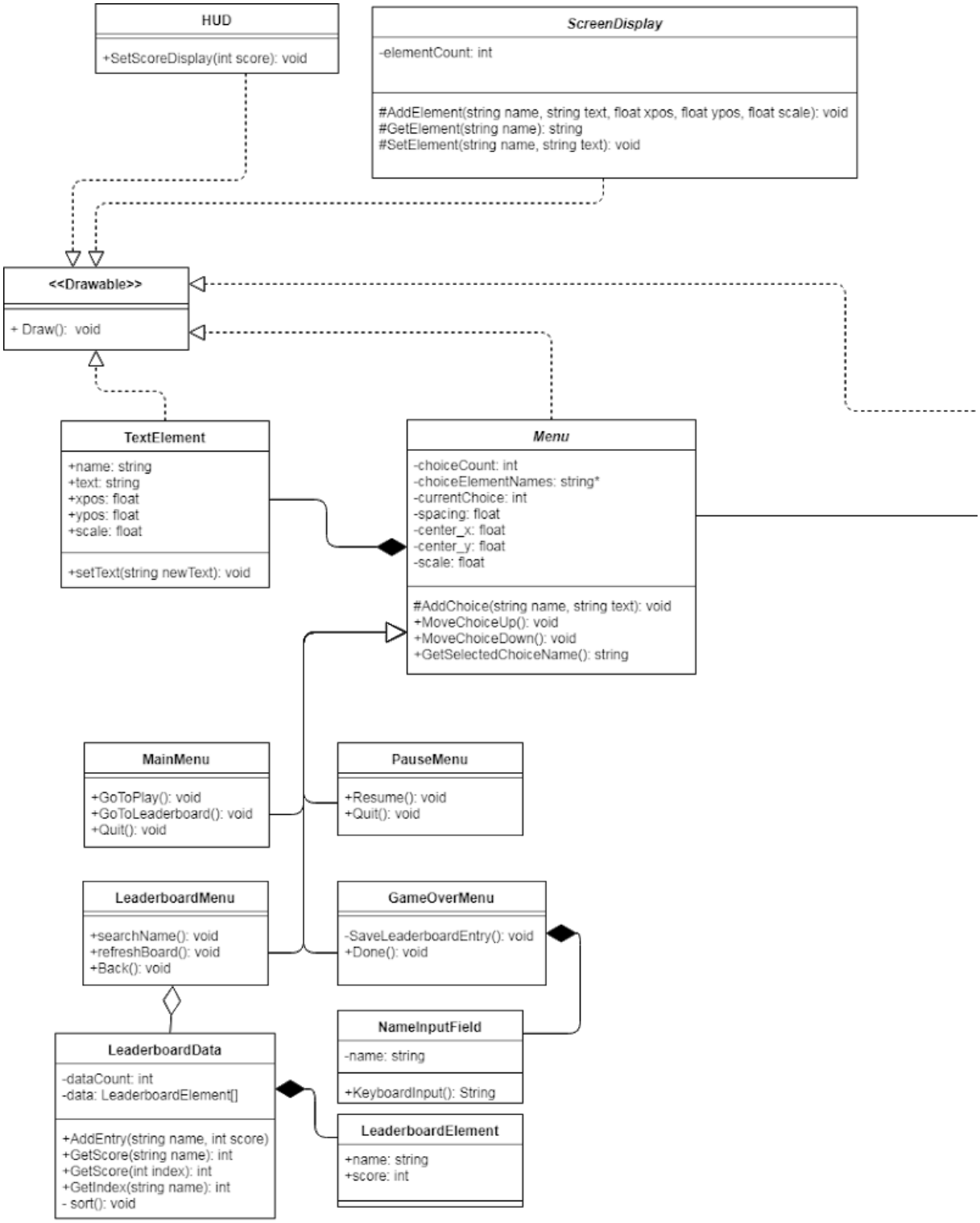
Vision/Description: We started our project with the vision of an Android Studio mobile game. We did not anticipate a steep learning curve with this framework, so we switched our plan into designing our own MVC and creating a simple Java shooter game. We had the idea of being able to move around with WASD and aim with the arrow keys, and then shooting with space bar. The bullets can kill enemies and gain score that will be updated with JDBC to our local mySQL database when the game ends. Collision is implemented correctly with walls and bullets, and having enemies chase the player.

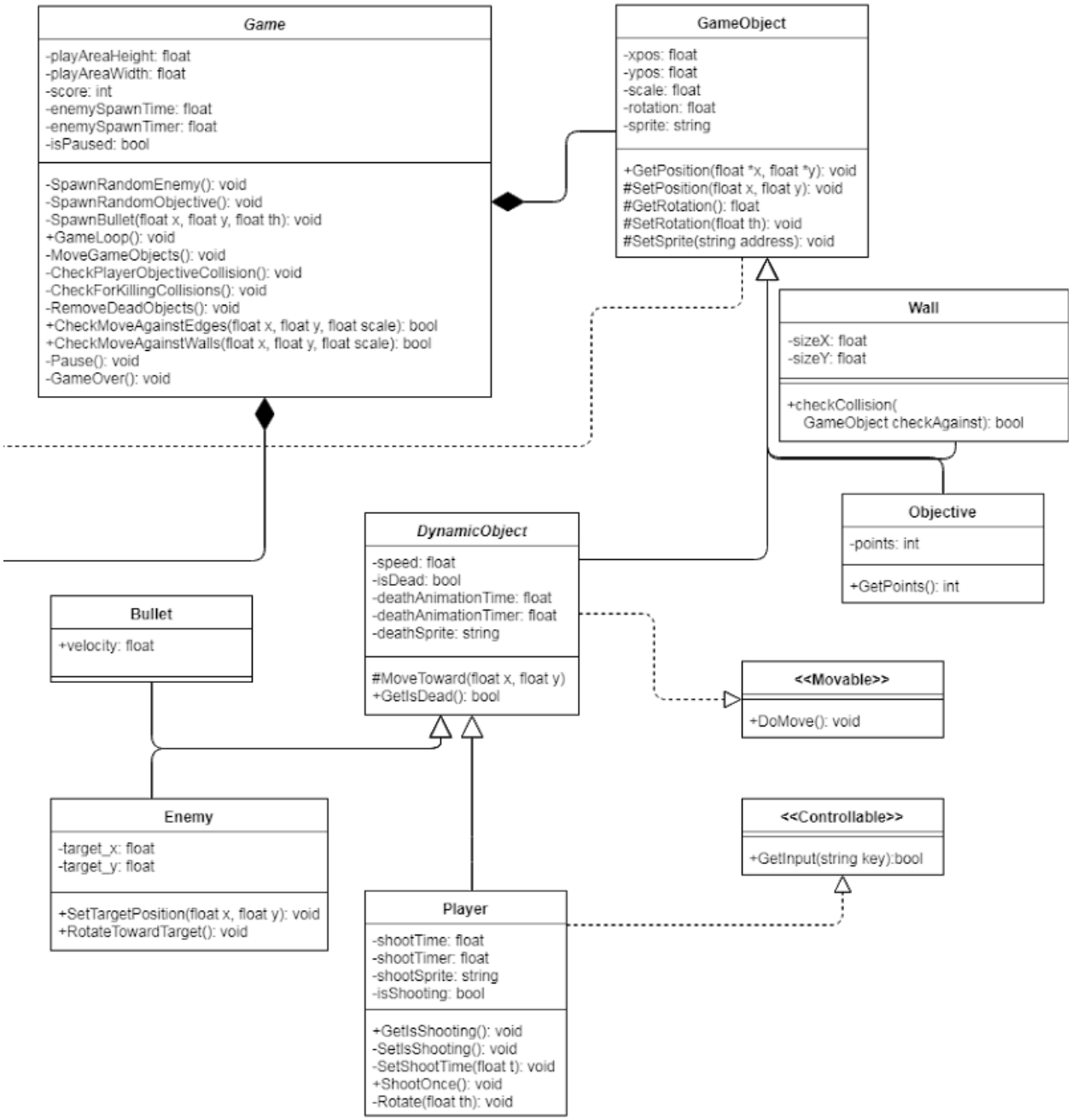
Implemented Features:
2. As a player, I can start the game
4. As a player, I can shoot projectiles
6. As a player, I can lose the game
7. As a player, I can gain points by killing enemies
9. As a player, I can move my character
13. Performance, the game should load in under 5 seconds

Features not implemented:
1. As an admin, I want to edit the leaderboards
3. As a player, I can pause the game
5. As a player, I can gain points by collecting objectives
8. As a player, I can quit the game
10. As a player, I can save my score with my name
11. As a player, I can search for other players' highscores
12. As a player, I can can search the leaderboard for a specific name
14. Performance, the game should run at least 60fps
15. Reliability, the leaderboard will always save and is always available

Note that many of the features left not implemented were specific goals for running on a phone. As we made the decision to switch platforms, these features became less of a necessity.

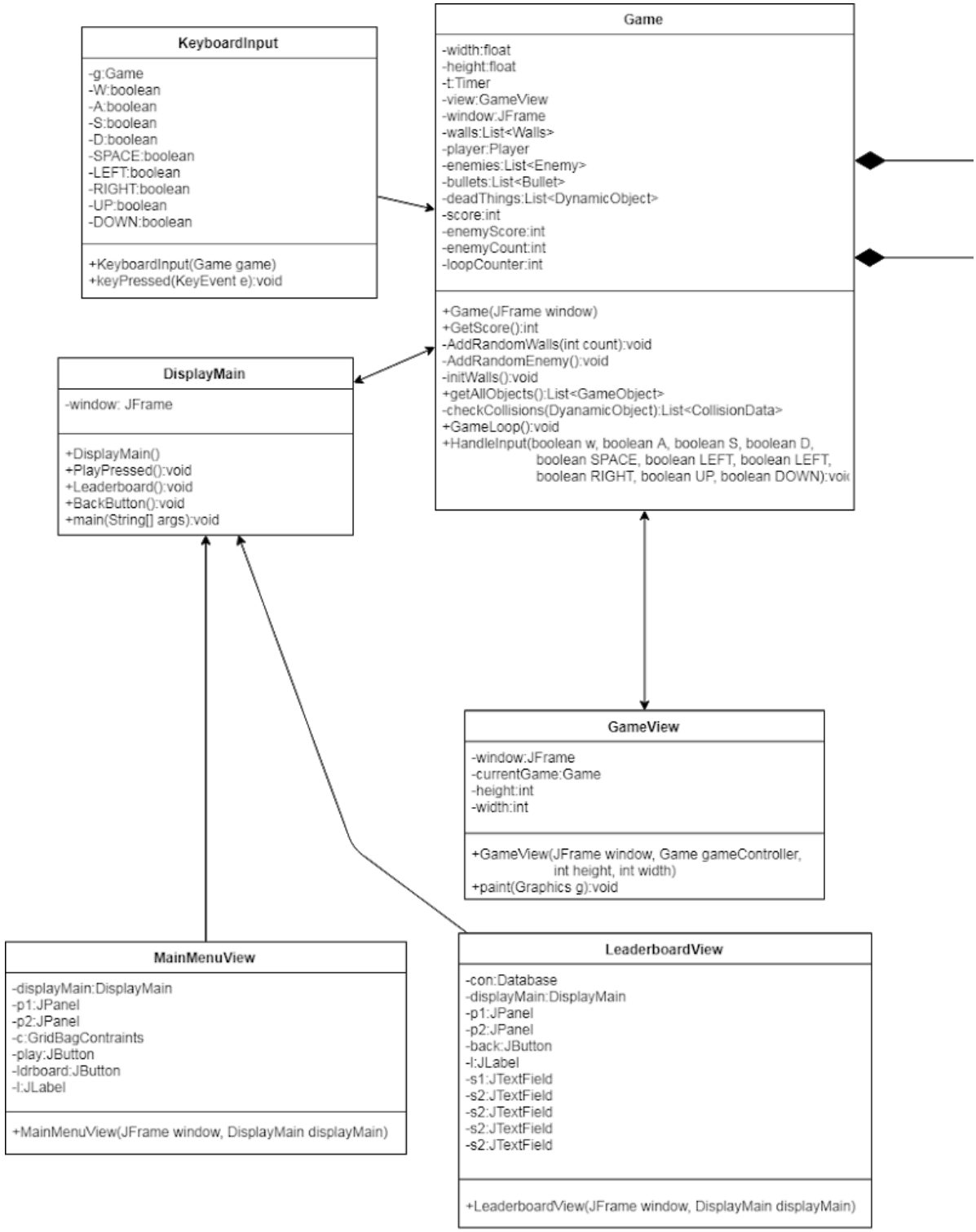
Part 2 class diagram

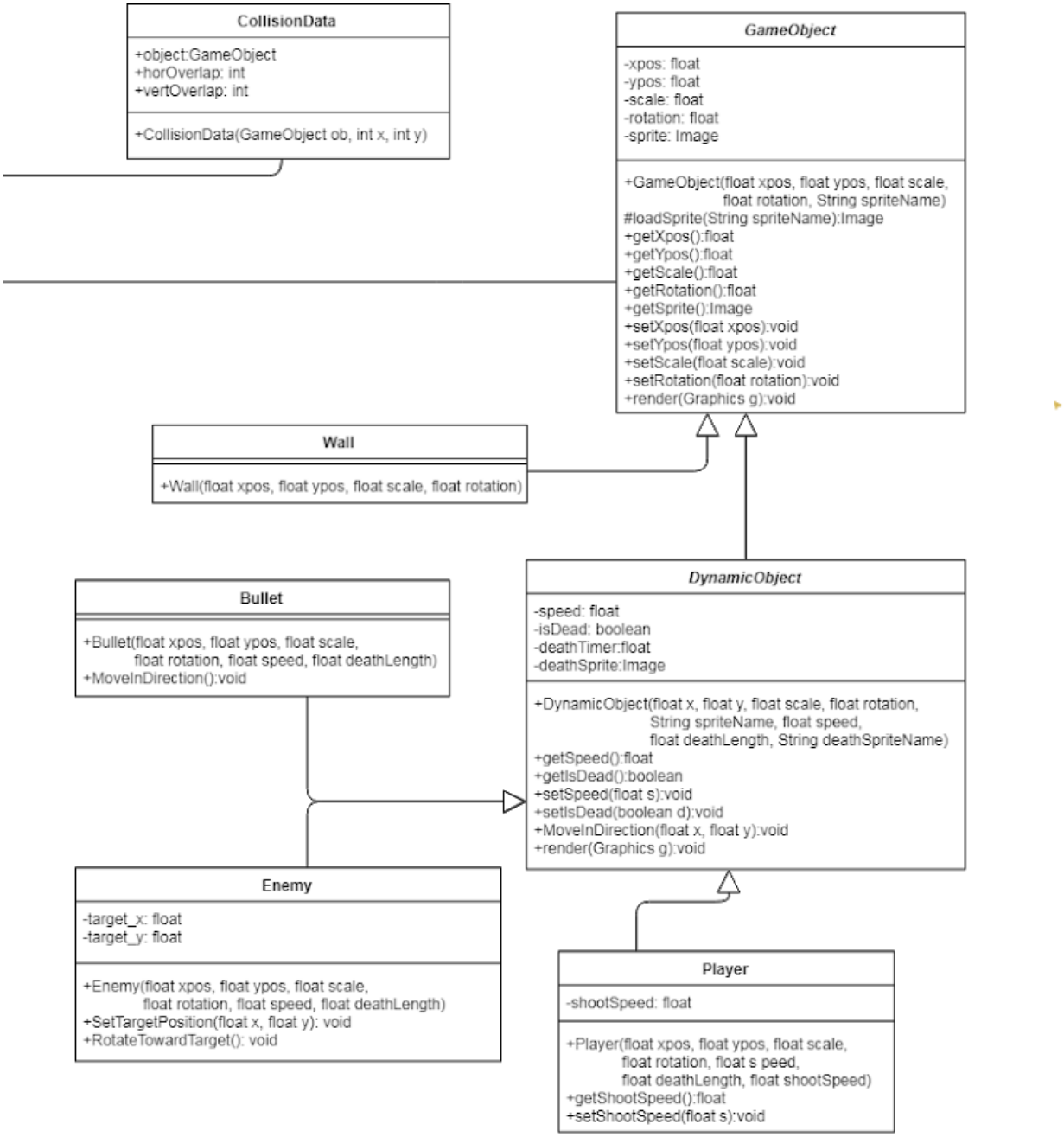




Full diagram can be found at:
https://drive.google.com/file/d/1VKFKGoq88hXPesJMecWb-FkF_66gLZDF/view?usp=sharing

Final diagram



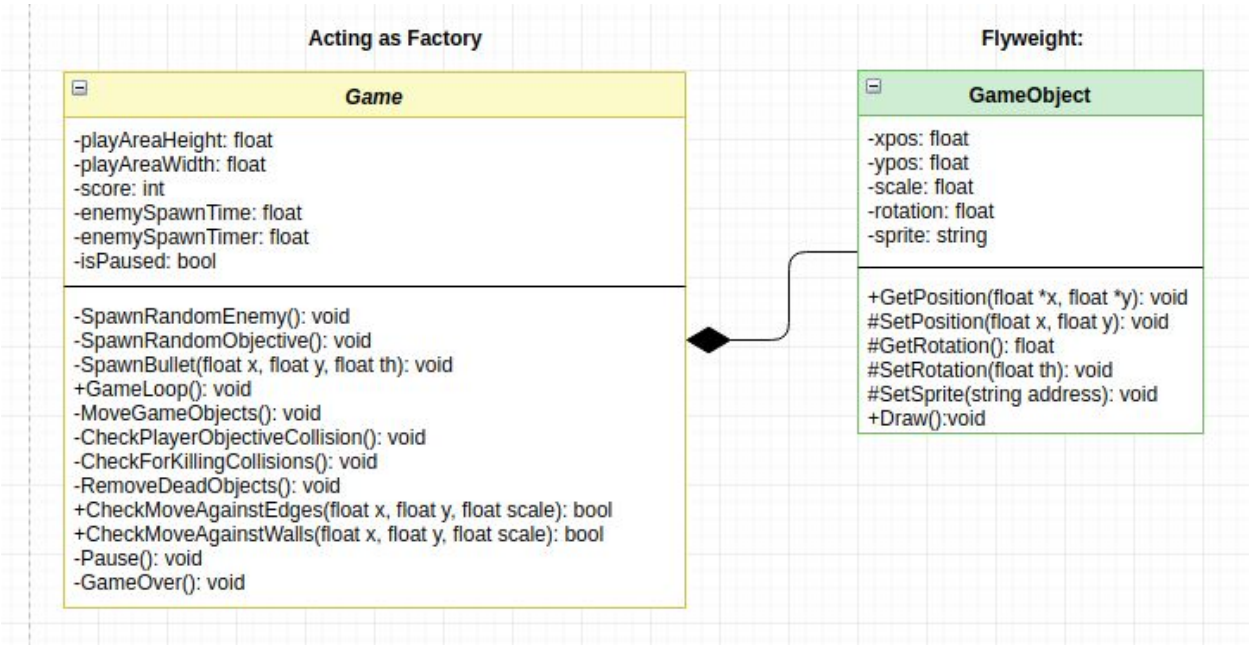


Full diagram can be found at:
https://drive.google.com/file/d/1CLHY_TZK2KPSFxiV0nlrfOoSAh3tEiW-/view?usp=sharing

What changed:
In terms of core design, the biggest change was the menu system. This is a consequence of changing from android studio to a custom MVC. As we developed we discovered that our predefined class attributes and functions were not able to fulfil our goals. We modified these attributes as we worked, but the core structure remained the same. A lot of the interfaces we originally had were not needed, as things like “drawable” were taken care of in java.swing.

Classes from class diagram that implement design patterns:

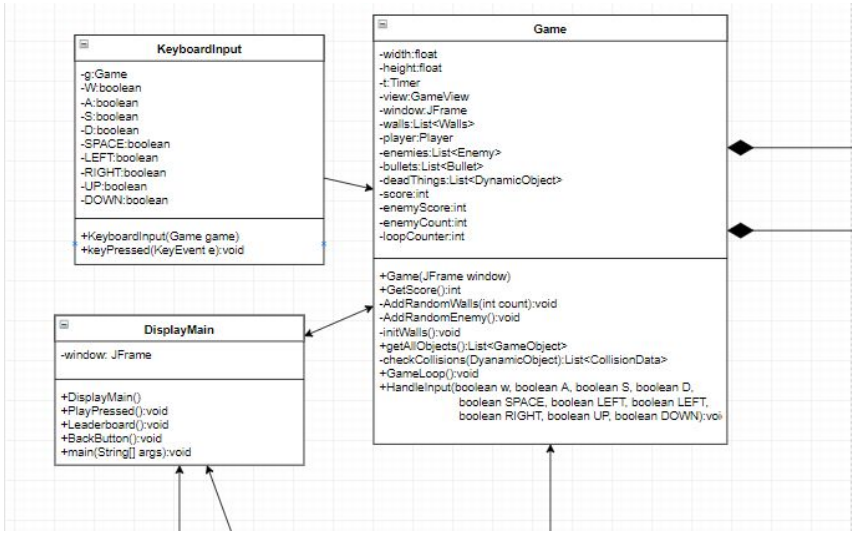
Flyweight:



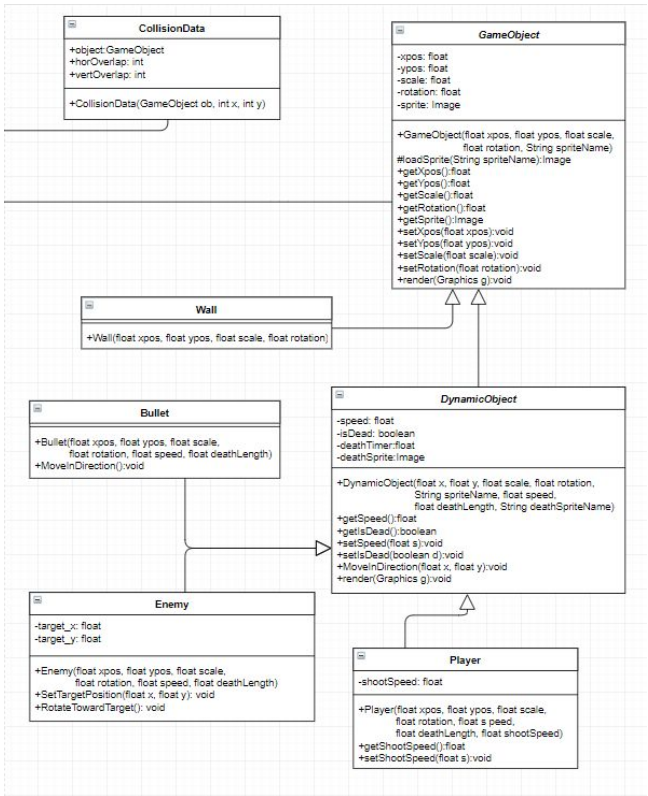
Model View Controller:

This design pattern can't be narrowed down to a specific part of the class diagram but is rather present in the diagram as a whole. The game accepts input through the controller which in turn modifies the models. The controller then updates the views with the information from the models.

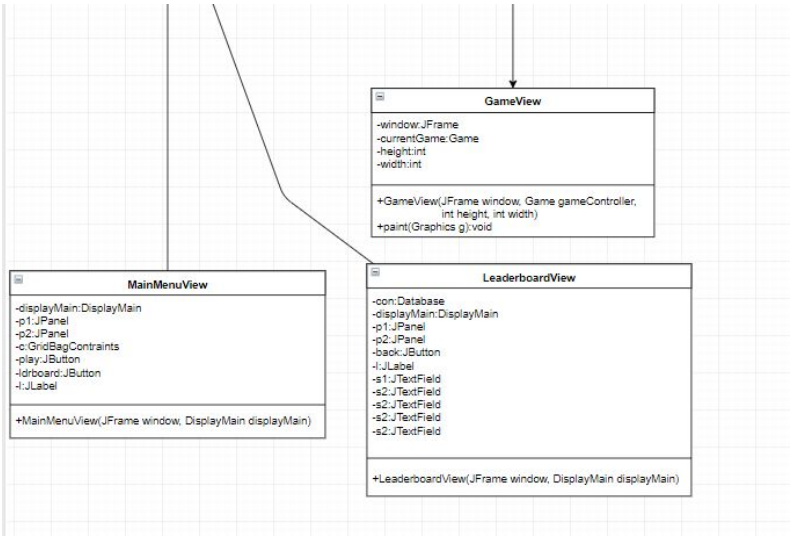
Controllers:



Models:



Views:



What we learned about designing a system: The model view controller design pattern is a really nice pattern to use, especially for games. The ability for models to be acted upon by controllers, and views being handled on their own allowed for easy implementation of the features we wanted. Turns out rendering graphics is very particular depending on the system you are running from; something we did not end up having the time to resolve. Knowing that we were going to use MVC from the beginning of the project would have drastically changed the way we laid out the classes originally, as most of the design was abandoned. Flyweight seemed to come automatically / naturally with the design of a game, which needs multiple entities of the same type.