Ultimate Earth Championship

SENG201 - 2023S1 Report

Farzad Hayatbakhsh - fha62 - 15424500
Oliver Coates - oco36 - 54134289

Project Structure:

From the project's conception, we deliberately set out to implement the Model View Controller architectural pattern, under the assumption that it would help to disentangle our logic from our GUI. The MVC pattern has been followed in this project with the exception of a few edge cases where it was decided to couple the model directly to the view for the sake of simplicity, such as the live match. We consider the implementation of the MVC architectural pattern a success which has helped keep the project's internal structure ordered and coherent across the eight weeks of development. The MVC can be seen in the class diagram, with each of the three boxes representing the Model, Controller and View.

We decided from the beginning that our game should be fun and enjoyable to play to best meet the requirements of the end user. An RPG, checkers style gameplay system was devised which would hopefully be simple to implement and understand while still remaining within the project specifications. We believe that the combat system has achieved these two goals.

Game logic itself typically follows a code exchange from the model classes through a singleton game manager which then passes information forward to the view, and then informs the model of any player input. We decided upon the singleton pattern for our game manager as having a globally accessible, single instance suited our need for a controller class. The actual game manager itself consists of either a 'graphical game manager' or a 'command line game manager' which both inherit from the game manager class and handle the task of controlling the game for their respective UI modes. The strategy pattern was used to order the display, which helped streamline and simplify development of functionality for both the GUI and CLI interfaces. We decided upon cutting the command line interface to focus on development of the GUI.

Each athlete (referred to in game and in code as 'champions') and weapons are defined explicitly in their own script, giving us direct control over their values, though they both inherit from either an athlete or weapon superclass which contains the object's logic. Both athletes and weapons implement the purchasable interface which allows for the view to easily interpret and show certain values relevant to buying and selling

Our unit test coverage across our whole project currently sits at: 50.5%. Our packages which contain the majority of game logic: Manager, Model, Events, Champion, Story and Weapons all have coverages greater than 90% with the exception of manager due to the code related to managing views which brings its coverage down to 75%. Packages which govern the GUI like views, display & match have a very low coverage in our unit tests. To remedy this lack of unit testing, **user testing** was conducted with the game design students of the game dev society, whose feedback was invaluable in detecting bugs and incorporating quality of life improvements.

This project has been very interesting. Learning Java, Swing, Junit and UML has certainly presented a number of learning challenges which we have had to overcome. We would both agree that this project has been put together in an intelligent manner which is challenging and time-consuming, while still being fun and open-ended. In future, perhaps the Software Engineering school could place more emphasis upon version control and AGILE practices earlier on in the course and encourage students to follow these to work towards a more stress free, better managed project.

Overall, we are both very happy with the outcome of Ultimate Earth Championship, not only as a project which fulfilled the requirements, but as a fun game and an exercise in good project management, AGILE and version control practices. Throughout development, we followed a workflow in which we would hold a code review every Friday in which the week's work was reflected on, code was peer-reviewed and planning made for the following week. After Friday we would work through our workload organized on a KANBAN board, meeting up every Wednesday morning to discuss progress, problems and code together. We made full use of git as our version control system, ensuring to branch with each new feature and merge every week or half week as was practical. The use of AGILE, a KANBAN board, weekly in-person meetups and git has been a huge success for this project and will most definitely be continued in future.

When our work system failed and problems began to emerge, it was mainly due to our group's inexperience in working with large systems. Our understanding of the Model View Controller has improved greatly throughout this project, and our lack of understanding at the beginning caused a multitude of design errors to lead to unnecessary high coupling. A prime example of this is the live match class, which rapidly grew from a fairly modest class, into its current, somewhat over-coupled state, over the period of around two weeks. A refactor of the live match class was planned for the last week of development, but we decided to cut it in favor of focusing upon reworking parts of the front facing graphical user interface.

In future, we will better adhere to certain architectural and design patterns that we wish to implement. In our current project, our game manager is becoming rather bloated from having to deal with communications between the entire model and entire view. To prevent this, in future projects we would like to create individual controllers for each of the views to better separate user input functionality from the model.

Time spent:
Oliver Coates (oco36) - 74 hours
Farzad Hayatbakhsh (fha62) - 70 hours

Agreed percentage of contribution:
Oliver Coates (oco36) - 50%
Farzad Hayatbakhsh (fha62) - 50%