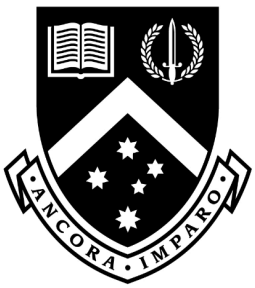


MONASH University

FIT3077 | Sprint 4 Report | S1 2024

Members: Gde Putu Guido, Timothy Suria, Andy Tay, Sayyidina Malcolm

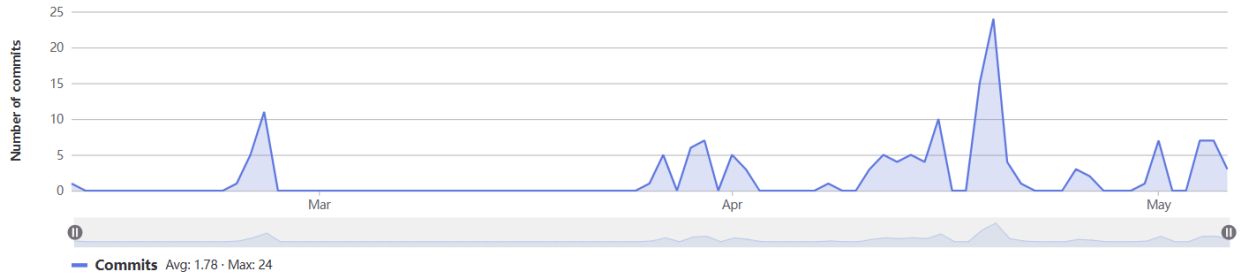
Authored by: Team 098

Tutorial: Tuesdays 12-2 pm

This assignment was made to provide a learning experience and understanding from our eyes of the subject of FIT3077 and also to fulfill the requirements of the assignment (submission). We do understand that there might be mistakes in writing, and we apologize for that.

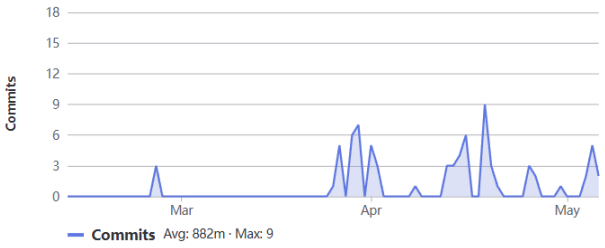
Contributor Analytics Screenshot

Excluding merge commits. Limited to 6,000 commits.



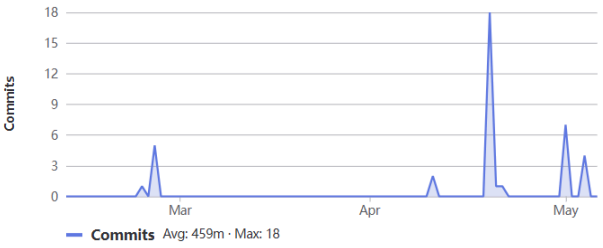
smal0039

75 commits (smal0039@student.monash.edu)



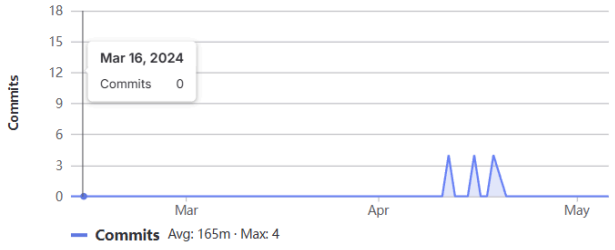
gdep0001

39 commits (gdep0001@student.monash.edu)



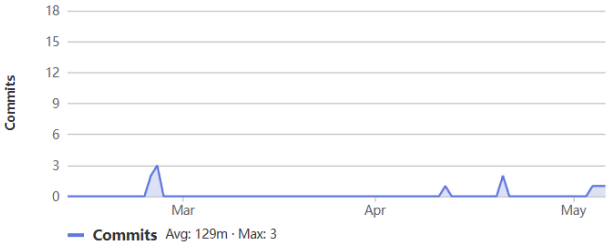
Timothy Suria

14 commits (126694900+timothysuria@users.noreply.github.com)



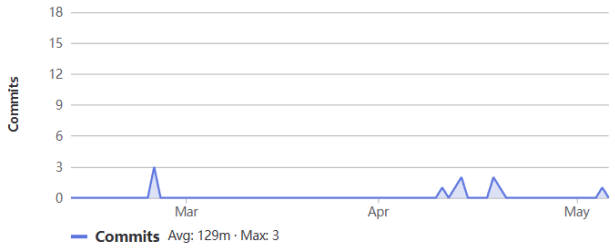
tsur0005

11 commits (tsur0005@student.monash.edu)



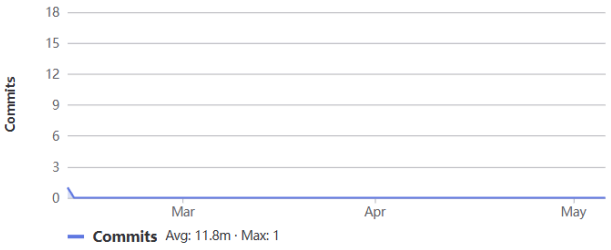
Andytay22

11 commits (111366969+andytay22@users.noreply.github.com)



Matt Chen

1 commit (matt.chen@monash.edu)



Self-Defined Extensions Explanation for Fiery Dragons Game

Notes & Pre-discussion notice

This section is dedicated to the self-defined extensions implemented to enhance the Fiery Dragons game. It is important to note that the extensions defined by the Sprint 4 briefing will not be fully explained here, as the assignment instructions did not indicate such a requirement. Therefore, the focus of this section will solely be on the “enhancements” (self-defined extensions) introduced by the team to enrich the gameplay experience.

Introduction

In enhancing the Fiery Dragons game, two significant self-defined extensions have been introduced to add depth, engagement, and variability to the gameplay. These extensions are the Stats Screen View and the Robin Pig (Thief) mechanic. Each extension is designed to enrich the player's experience by introducing new features that complement the base game's mechanics. This document provides a comprehensive description of these extensions, detailing their functionality, implementation, and impact on the game.

Stats Screen View

The Stats Screen View extension is designed to provide players with detailed statistics about their gameplay. This feature enhances the game by offering insights into player performance, fostering a competitive environment, and encouraging players to improve their skills. The key components and functionalities of the Stats Screen View are outlined below:

- **Loading Game State**: The Stats Screen View loads the saved game state from a specified file. This involves deserializing the game data to reconstruct the game state, including player statistics and game progress.
- **Player Statistics Extraction**: The extension extracts various statistics for each player, such as the total steps taken, backward steps, and wrong guesses. These statistics are crucial for assessing player performance and are displayed in a tabulated format.
- **User Interface**:
    - **Table View**: The primary interface element is a TableView that displays player statistics. The table includes columns for player names, total steps, backward steps, and wrong guesses. Each column is styled for better readability.
    - **Labels and Buttons**: Additional UI elements include labels indicating the number of players and a back button to return to the main menu. These elements enhance the user experience by providing necessary information and navigation options.
    - **Styling**: The UI components are styled with appropriate colors and alignments to ensure a visually appealing and user-friendly interface.
- **Functionality**:
    - **Sorting Players**: Players are sorted based on their total steps in descending order to highlight the most active players.
    - **Scene Management**: The Stats Screen View manages the transition between the stats screen and the main menu, ensuring smooth navigation.

Code Implementation:

- **StatsScreenView Class**: This class manages the stats screen's display and functionality. It includes methods for loading game state, extracting and displaying player statistics, and handling UI interactions.

Stats Screen View, why it is a human-centric extension

Based on what the team have learned throughout the journey in FIT3077 so far, up until now, the Stats Screen View is considered a human-centric extension because it focuses on enhancing the user experience by providing meaningful feedback and insights into player performance. By offering statistics, players can understand their gameplay information, track their progress, and they could identify areas for improvement. This feature promotes a more engaging and accessible experience, as players are motivated to perform better and achieve higher scores, fostering a sense of achievement and satisfaction.

Robin Pig (Thief) Mechanic

The Robin Pig (Thief) mechanic introduces an element of chance and strategy to the game. This extension allows players to encounter a Thief (Robin Pig) during gameplay, which can either move them forward or backward by three steps. The key components and functionalities of the Robin Pig mechanic are outlined below:

- **Thief Character**: The Robin Pig is a new animal type classified as a Thief. It is represented by a unique image and is added to the game board like other animal cards.
- **Gameplay Mechanic**:
    - **Encountering the Thief**: When a player encounters a Robin Pig on the game board, a special mechanic is triggered.
    - **Chance Element**: The outcome of encountering a Robin Pig is determined by chance. The player either moves forward three steps or backward three steps, adding an unpredictable element to the game.
- **Functionality**:
    - **Random Movement**: The direction of movement (forward or backward) is decided randomly using a boolean flag.
    - **Handling Movement**: The game state is updated based on the outcome, with appropriate messages displayed to indicate the result of the encounter.

Code Implementation:

- **RobinPig Class**: This class extends the Animal class, defining the Robin Pig as a Thief with specific attributes and image resources.
- **GameStateController Enhancements**:
    - **isRobinCell Method**: Checks if the player's current position contains a Thief cell.
    - **handleRobinCard Method**: Handles the logic for moving the player forward or backward based on the random outcome.

Conclusion

Our team hopes that the addition of the Stats Screen View and the Robin Pig (Thief) enhances the Fiery Dragons game by providing players with detailed performance metrics and introducing an an element of chance. These extensions are meticulously designed and implemented to integrate with the existing game mechanics, offering an enriched and engaging gameplay experience. By tracking player statistics and adding variability through the Thief encounters, these extensions contribute to a more dynamic and competitive environment, encouraging players to strategize and improve their gameplay.

Reflection on Sprint 3

Reflecting on Sprint 3, our team encountered various levels of difficulty in incorporating the necessary elements into the design and implementation of the Fiery Dragons game. Here, we will discuss the difficulty levels of each aspect, outline the relevant design and implementation factors, and suggest improvements for easier future integration.

1. Review of Sprint 2 Tech-based Software Prototypes

Difficulty Level: Moderate

Challenges:

- **Establishing Assessment Criteria:** Defining comprehensive assessment criteria based on the ISO/IEC 25010 quality model was challenging. It required thorough understanding and agreement among team members.
- **Collaborative Review:** Reviewing each team member's prototype impartially and constructively was time-consuming but crucial for the success of Sprint 3.

Sprint 3 Design Aspects:

- **Positive:** The structured approach to reviewing each prototype helped in identifying the best elements to incorporate into the final design.
- **Negative:** The lack of prior standardized assessment metrics made it harder to objectively compare prototypes.

What I Would Change:

- If we had the opportunity to start over, we would prioritize the establishment of standardized assessment criteria much earlier in the project timeline. By doing so, we would streamline the review process and ensure that all team members' prototypes are evaluated consistently and fairly. This approach would not only save time but also enhance the clarity and objectivity of our assessments, leading to more accurate and reliable feedback. Additionally, having these criteria in place from the beginning would facilitate better alignment among team members, reducing misunderstandings and fostering a more cohesive and efficient development process.

2. Consolidation of Prototype Elements

Difficulty Level: High

Challenges:

- **Merging Different Approaches:** Integrating the best elements from each prototype into a single, cohesive solution direction was complex. Each prototype had unique strengths and weaknesses that needed careful consideration.

- **Conflict Resolution:** Resolving conflicts between different design philosophies and coding styles required extensive discussion and compromise.

Sprint 3 Design Aspects:

- **Positive:** The modular design of most prototypes allowed for easier integration of individual components.
- **Negative:** Inconsistent coding standards and architectural patterns across prototypes led to integration difficulties.

What I Would Change:

- I would enforce a common coding standard and design pattern usage from the beginning of Sprint 1. This would ensure consistency and reduce integration issues in later stages.

3. Development of CRC Cards and Class Diagram

Difficulty Level: Moderate

Challenges:

- **Detail and Accuracy:** Creating detailed CRC cards and a comprehensive class diagram required a deep understanding of the consolidated design. Ensuring accuracy and completeness was challenging but essential.
- **Alternative Designs:** Considering and rejecting alternative distributions of responsibilities added to the complexity.

Sprint 3 Design Aspects:

- **Positive:** The modular and well-defined responsibilities of each class helped in creating accurate CRC cards and class diagrams.
- **Negative:** The absence of a unified approach to responsibility distribution initially made the process more difficult.

What I Would Change:

- Introducing collaborative design sessions early on would help align team members on responsibility distribution. This would make creating CRC cards and class diagrams more straightforward.

4. Implementation of the Tech-based Software Prototype

Difficulty Level: High

Challenges:

- **The complexity of Game Mechanics:** Implementing the full game functionality, including player movements, chit-card interactions, and game state management, was complex.

- **Ensuring Functionality:** Ensuring that the implemented prototype adhered to all game rules and was fully functional required rigorous testing and debugging.

Sprint 3 Design Aspects:

- **Positive:** The use of the Model-View-Controller (MVC) pattern in Malcolm's prototype provided a strong foundation for separating concerns and managing complexity.
- **Negative:** The presence of code smells, such as tightly coupled classes and scattered state management, made the implementation more error-prone and difficult to debug.

What I Would Change:

- Implementing design patterns like Singleton for game state management and ensuring loose coupling between classes from the start would simplify the implementation and enhance maintainability.

5. Creation of the Executable and Video Demonstration

Difficulty Level: Moderate

Challenges:

- **Executable Creation:** Creating a reliable executable with clear installation instructions required meticulous attention to detail.
- **Video Presentation:** Coordinating the video demonstration to cover all key functionalities within the time limit was challenging, especially ensuring all team members contributed effectively.

Sprint 3 Design Aspects:

- **Positive:** The detailed planning and documentation helped in creating a functional executable and coherent video presentation.
- **Negative:** The lack of initial planning for the video presentation structure led to a disjointed demonstration and exceeded time limits.

What I Would Change:

- Establishing a clear script and presentation flow for the video early in the process would ensure a more polished and time-efficient demonstration.

Conclusion

Overall, Sprint 3 provided valuable insights into the complexities of integrating diverse elements into a cohesive software solution. By addressing the identified areas for improvement, future sprints could achieve smoother integration and more efficient implementation of extensions. Implementing standardized assessment criteria, enforcing consistent coding standards, and adopting robust design patterns early on would significantly enhance the development process.

Enhancing and Deepening the Reflection

We are going to enhance and deepen the reflection on our Sprint 3 design and implementation by reiterating through the various aspects of extensibility. This will involve a thorough review of the level of difficulty encountered for each incorporated extension, as well as a critical assessment of our own practices. We aim to identify meaningful improvements for future practice and outline a range of strategies and techniques that could be employed to achieve better outcomes. This approach ensures that our reflection is comprehensive and not redundant, providing valuable insights for future development.

Overall Extensibility of the Design

Our design for Sprint 3 demonstrated significant extensibility, particularly in the modular architecture we adopted. The use of the Model-View-Controller (MVC) pattern allowed for clear separation of concerns, making it easier to integrate new features without disrupting existing functionality. This was evident in how smoothly we incorporated the Stats Screen View and the Robin Pig extension. However, the extensibility was occasionally hindered by tight coupling in certain parts of the code, which we need to address in future iterations.

Executable Description and Instructions for Fiery Dragons Game

Introduction

Our group created this section to provide a clear description of the executable for our Fiery Dragons game, detailing the platforms it runs on, instructions for running the game, and even the process for creating the executable from the source code. We hope that unlike sprint 3 and based on the received feedback, these instructions will guide readers and markers and maybe developers in successfully deploying and running the game on their systems.

Platforms

The Fiery Dragons game has been developed and thoroughly tested in a MacOS environment. While it is expected to be compatible with other operating systems such as Windows and Linux, users on these platforms should perform their own tests to ensure compatibility and smooth operation.

Running the Executable

To run the Fiery Dragons game executable, follow these steps:

1. **Ensure Java is Installed**: This may be trivial but it is paramount that you make sure you have Java installed on your system. You can download the latest version of Java from the official Oracle website or use a package manager to install it.
2. **Download the Executable JAR File**: Obtain the `FieryDragon.jar` file, which is the compiled executable of the game. This file can be provided by the developers or built from the source code as described later in this document.
3. **Open a Terminal or Command Prompt**:
    - On **MacOS** or **Linux**, you can open the Terminal.
    - On **Windows**, you can open Command Prompt or PowerShell.
4. **Navigate to the Directory Containing the JAR File**: Use the `cd` (change directory) command to navigate to the folder where the `FieryDragon.jar` file is located. For example:
    - ***cd path/to/jar/file***
5. Run the JAR File: Execute the JAR file using the following command:
    - java -jar FieryDragon.jar
6. Game Launch: The game should now launch, and you will be able to start playing the Fiery Dragons game.

Creating the Executable from Source Code (the method that our team uses)

To create the executable JAR file from the source code, our team utilized this set of steps:

1. **Ensure Java Development Kit (JDK) is Installed**: You need the Java Development Kit (JDK) to compile the source code. You can download it from the official Oracle website or use a package manager to install it.

2. **Download and Set Up the Source Code**: Obtain the source code for the Fiery Dragons game. This can be done by cloning the repository from a version control system like Git:

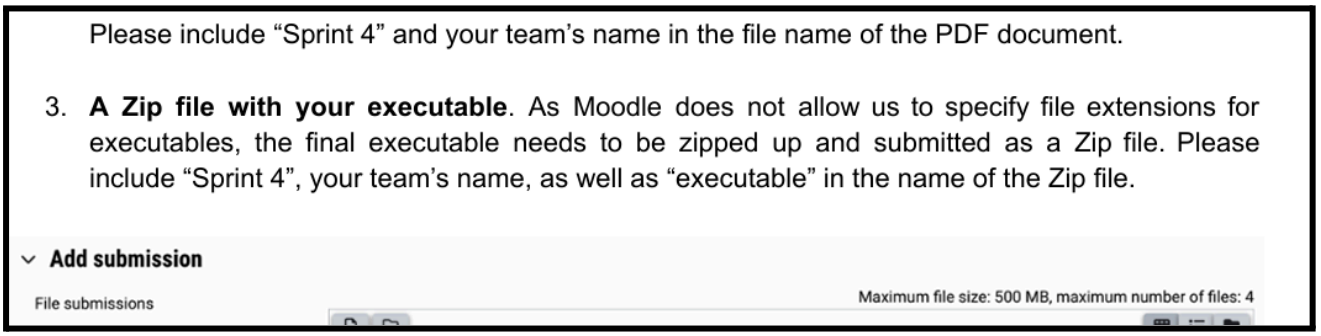
   ```
   git clone <repository-url>

   cd <repository-directory>
   ```

3. **Compile the Source Code**: Our group uses Mavn to compile the source code and create the JAR file. Below are the steps for using Maven:
   - Again, this may be trivial, for the sake of explanation's completeness, we will mention it, Ensure you have Maven installed. You can download it from the official Maven website or use a package manager.
   - Navigate to the directory containing the `pom.xml` file (the Maven configuration file).
   - Run the following command to compile the code and create the JAR file:
     ```
     mvn clean package
     ```
   - The compiled JAR file will be located in the `target` directory, named `FieryDragon.jar` (our team typically name it this way).
4. **Build Artifacts**: **This is the method that our team uses, it is also build in inside Intellij,** which is the IDE that we used since the beginning of the semester (JAVA). The build artifacts in Java are managed using build tools like Maven, which handle dependencies and project configurations. Ensure that all dependencies are correctly specified in the `pom.xml` (for Maven).
5. **Run the JAR File**: After creating the JAR file, you can run it using the same steps described in the "Running the Executable" section.

Additional explanation, steps on how our group created the executable using Inteliij build0in

Conclusion

Our team believes that by following these instructions, users and markers can successfully run and create the Fiery Dragons game executable, **even though we will still provide it in a zip file format as the Sprint 4 briefing states (we will attach a screenshot of this from the assignment briefing):**


Please include "Sprint 4" and your team's name in the file name of the PDF document.

3. **A Zip file with your executable**. As Moodle does not allow us to specify file extensions for executables, the final executable needs to be zipped up and submitted as a Zip file. Please include "Sprint 4", your team's name, as well as "executable" in the name of the Zip file.

Add submission

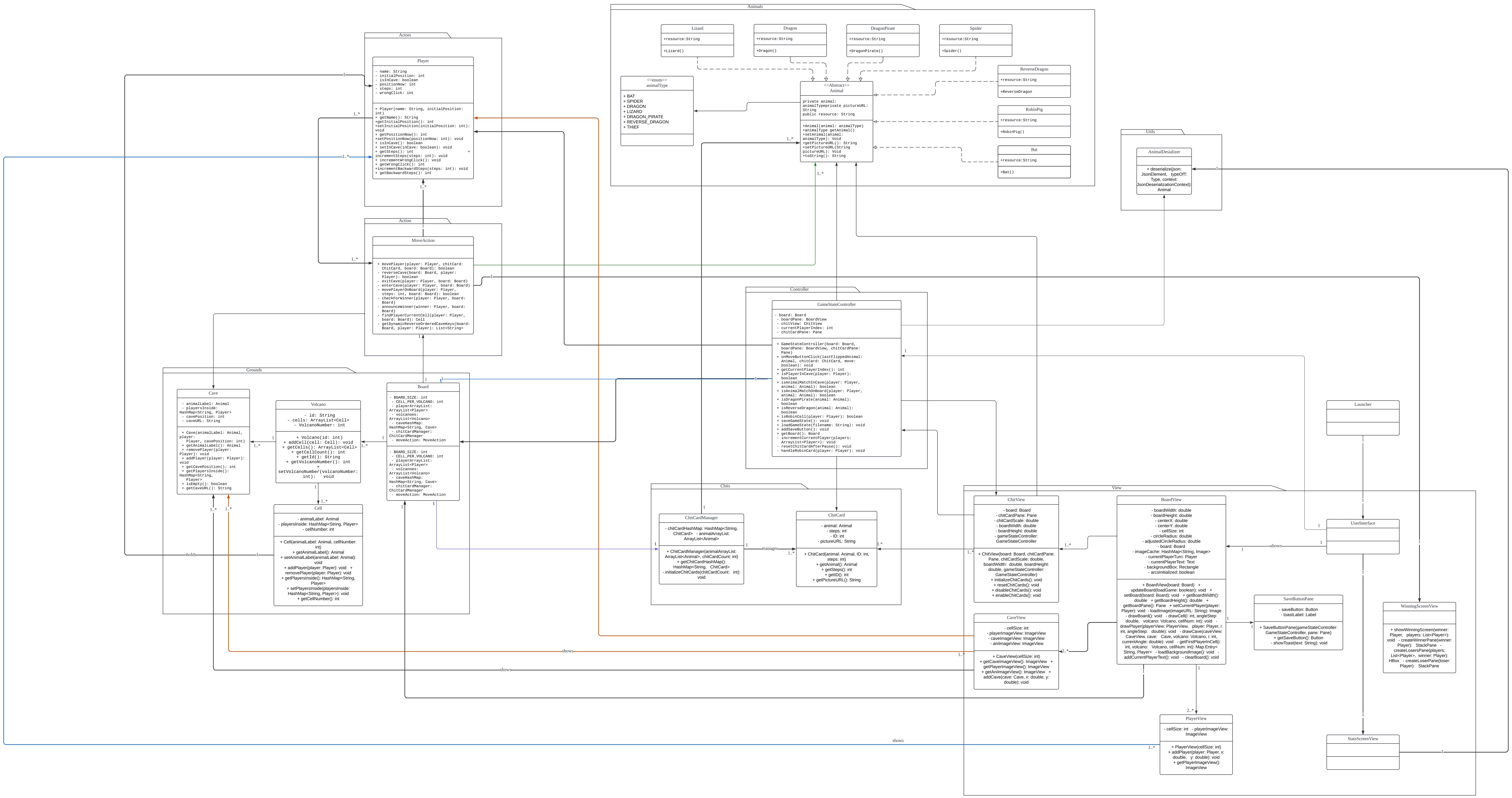
File submissions

Maximum file size: 500 MB, maximum number of files: 4

This explanation section will provide the required explanation of our executable, the platforms it runs on, how it needs to be run, as well as instructions how it can be created from source code.

Our group have tried to make sure that the game has been designed to be easily deployable on various platforms, ensuring a smooth and enjoyable gaming experience. The detailed steps provided here will help in

navigating the process of running and building the game from the source code, fostering a better understanding of the game's deployment and execution.


Actors
Player
Action
MoveAction
Grounds
Cave
Volcano
Board
Cell
holds
Animals
Lizard
Dragon
DragonPirate
Spider
ReverseDragon
RobinPig
Bat
<<enum>>
animalType
<<Abstract>>
Animal
Utils
AnimalDesializer
Controller
GameStateController
Chits
ChitCardManager
ChitCard
manages
View
ChitView
BoardView
Launcher
UserInterface
SaveButtonPane
WinningScreenView
CaveView
PlayerView
StatsScreenView
shows