MONASH University

FIT3077 - Architecture and Design

Sprint One

MA_Tuesday08am_Team123

Members:

Maliha Tariq

Loo Li Shen

Khor Jia Wynn

Dizhen Liang

MONASH University

Table of Contents

MONASH University

1. Team Information

1.1 Project Summary

This project involves developing a game ‘Fiery Dragons’. The game needs to adhere to proper software development practices and object-oriented design principles.

At the onset of development, our focus lies on the implementation of the core functionalities essential to the Fiery Dragons gaming experience. This involves crafting a robust framework that enables two to four players to engage in gameplay within a unified client instance. Paramount to our efforts is the meticulous alignment of the game mechanics with the standard rules, ensuring a cohesive and immersive gameplay experience for all participants.

This report comprehensively covers various aspects of our project, including the technology stack to be used, the user stories, the domain model, and the low-fidelity (low-fi) prototype design.

1.2 Professional Team Name and Professional Team Photo

Team Name: “Swift Crafters”

Team Photo:

MONASH University

1.3 Team Members Information

Member 1

Name: Maliha Tariq

Student ID: 33473692

Email address: mtar0012@student.monash.edu

Technical Strengths & Professional Strengths: Java, Python and CSS

Fun Fact: I am deathly scared of cats

Member 2

Name: Loo Li Shen

Student ID: 32619685

Email address: lloo0011@student.monash.edu

Technical Strengths & Professional Strengths: Java, Python

Fun Fact: I have 200 myr to my name

Member 3

Name: Khor Jia Wynn

Student ID: 33033919

Email address: jkho0044@student.monash.edu

Technical Strengths & Professional Strengths: Java, Python

Fun Fact:

Member 4

Name: Liang Dizhen

Student ID: 31240291

Email address: dlia0009@student.monash.edu

Technical Strengths & Professional Strengths: Java, Python

Fun Fact:

1.4 Team Schedule

Our team follows a structured schedule to ensure efficient collaboration and productivity. We have weekly meetings every Tuesday during our designated workshop session as well as on Fridays (discord meeting). This serves as an opportunity for discussions on ongoing tasks, project updates and upcoming deadlines. During these meetings, we collectively review the work at hand and allocate responsibilities among team members based on individual strengths and expertise. Once tasks are distributed, each team member works on their assigned portion. We conduct regular check-ins to review each other's work. This approach ensures the quality of our work and fosters a sense of accountability and support within the team.

To keep track of our work within the team, we use Google Sheets as our primary tool for recording individual contributions. We have a shared spreadsheet where each team member logs their tasks, progress, and any relevant notes or updates

1.5 Technology Stack and Justification

Programming Language: Java

Justification: Our team has collectively decided to use Java as the primary programming language for developing the "Fiery Dragons" game. This decision is primarily based on our team's experience and proficiency with Java. Java's object-oriented programming (OOP) capabilities make it an ideal choice for designing and implementing the complex interactions and behaviours required in a game environment. Furthermore, Java's platform independence ensures that our game can be run on any operating system without the need for modification, thereby broadening our potential user base.

User Interface API: Swing

Justification: For the game's user interface development, we have chosen the Swing library. Swing is a part of Java's standard library, which provides a set of widgets and tools for creating graphical user interfaces (GUIs). Though our team is not familiar with Swing, its comprehensive documentation and support available online make it a good choice for developing the game's interface. Swing's integration with Java also facilitates a smoother development process, allowing us to focus on game logic rather than compatibility issues.

IDE: IntelliJ IDEA

Justification: The Integrated Development Environment (IDE) selected for this project is IntelliJ IDEA. Our choice is motivated by IntelliJ's advanced code completion, refactoring, and debugging capabilities, specifically tailored for Java development. IntelliJ IDEA supports direct integration with Swing, which simplifies the GUI design process through its intuitive drag-and-drop interface builder. The majority of our team members are already proficient with IntelliJ IDEA, which we anticipate will streamline the development workflow and enhance productivity.

Version Control: Gitlab

Justification: To manage our project's source code and facilitate collaborative development, we will be using Gitlab as our version control system. It will enable our team to track changes, revert to previous versions if necessary, and manage different development branches effectively.

Anticipated Support Needs:

Swing Advanced Features: While our team is confident in our ability to utilise Swing for basic GUI development, we may require support in implementing more advanced features or optimising the game's performance.

Game Design Principles: As we are more experienced with general software development than game development specifically, guidance on best practices and design patterns specific to game development would be beneficial.

Testing and Debugging: Support in setting up a robust testing framework for game development could help ensure the reliability and quality of the game.

Final Justification:

Our choice of technology stack is grounded in our team's current expertise and the specific requirements of the "Fiery Dragons" game. Java, with its powerful OOP features, combined with Swing for GUI development, provides a solid foundation for creating an interactive game. This technology stack not only leverages our existing skills but also aligns with the project's needs, ensuring a smooth development process and a high-quality final product.

1.6 Analysis of Alternatives

We considered Python with Pygame as a viable alternative to our primary choice of Java and Swing for the development of "Fiery Dragons." Python offers simplicity and readability, which could facilitate faster development cycles. Pygame, a library tailored for video game development in Python, provides essential tools for creating interactive games, making it suitable for beginners and advanced developers.

While Python is a powerful and flexible language that supports object-oriented programming (OOP), its dynamic nature can introduce complexities in enforcing strict OOP principles compared to Java. Java was designed with a strong emphasis on OOP, making it inherently structured to support complex object-oriented designs. Its syntax and language features encourage the use of interfaces, abstract classes, and strong typing, facilitating the implementation of design patterns and OOP principles straightforwardly.

Python's dynamic typing and flexibility, though advantages in many scenarios, can make it more challenging to apply strict OOP methodologies. The dynamic type system allows for more flexibility in how objects and classes interact, but this can lead to less predictability and more difficulty in ensuring type safety and adherence to specific OOP conventions. The choice between these two sets of technologies depends on the specific needs of our game, the team's familiarity with the programming languages, and the target platforms for the game.

2. User Stories

1. As a game designer, I want to have a game board with 8 segments and 4 characters so that players can experience diverse environments and choose from a variety of animal characters.

2. As a player, I want my game to start with my token placed at the cave, allowing me to plan and execute strategic moves to navigate back to the cave, thus fulfilling the game's primary objective.

3. As a player, I want a unique coloured animal to differentiate my game character from others.

4. As a player, I want my turn to end if the revealed chit card shows a different animal than my current state.

5. As a player, I want to be able to select chit cards where the animal depicted matches the animal in my current position on the game board so that I can strategically move forward in the game.

6. As a game designer, I want to cover up all the dragon cards that were flipped at the end of a player’s turn, to implement the memory component of the game.

7. As a player, I want to be able to reveal another chit card after a successful move to potentially move again.

8. As a game designer, I want to be able to shuffle the deck of chit cards and place them down in the centre.

9. As an admin, I want to provide clear guidelines for the game rules to ensure a fair and enjoyable experience for all the players.

10. As a game designer, I want to enforce a rule that prevents players from using the same chit card more than once during their turn, ensuring all players adhere to fair play standards.

11. As a game designer, I want to check if a player reaches their cave with an exact number of moves so that I can end the round and declare the winner.

12. As a game designer, I want the game to stop a player from moving its token when the token tries to gain a position that is already occupied by another player's token.

13. As a game designer, I want to implement the dragon pirate card to give the player the option of moving backwards.

14. As a parent, I want the game to offer educational benefits, such as memory skills and decision-making, for my child.

15. As a game designer, I want to shuffle the uncut segments cut segments as long as the cut and uncut segment

Extensions

16. As a player, I want the option to play with dummy players controlled by the game that follows the rules of the game.

17. As a player, I want to celebrate and congratulate the winner of the game, whether it's myself or another player, to acknowledge their achievement.

18. As a player, I want to have different game modes/versions of the game. For example, like in Monopoly where a specific location can grant you cards which gives your character effects whether it be good or bad, to make the game more interesting.

19. As a player, I want the chit cards to be shuffled randomly after every 2 rounds of players' movement, so all the players can not gain benefits through memorising the chit cards and more fun to play.

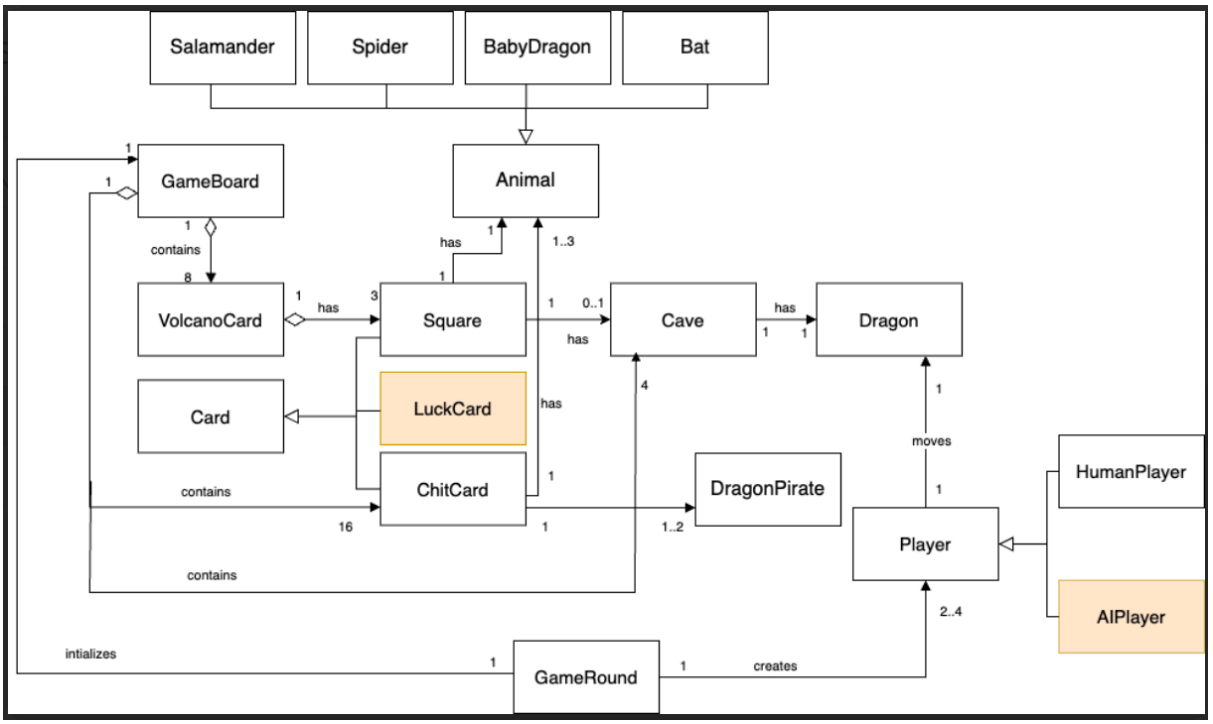
20. As a player, I want the luck card to allow me to switch the position with random players which is decided by the luck card, so more fun that the leader would not always take the lead.

21. As a player, I want the game to continue even if there is already a winner if playing with friends, but immediately end the game if playing solo.

22. As a player, I want to draw a luck card every round to hold, which can give me benefits when I use it in my own turn, so that would be more fun than just flipping the chit cards.

23. As a player, I want each character to have their inherent abilities to use to gain benefits, which would be more fun to play the game.

3. Domain Model



Design Rationale

GameRound, GameBoard

The GameRound entity models a game session, which involves two main entities: the players and the game board. Therefore, we have association relations between GameRound and GameBoard, as well as between GameRound and Player. This design choice aims to designate a main controller class to initialise the game components.

To render the board, we require user input regarding the number of players. For example, if the user chooses a 2-player mode, only two caves and dragon tokens are required. By defining the GameRound entity, the GameBoard entity does not need to be concerned with managing user input, instead shifting this responsibility to the GameRound entity (dependency inversion principle).

The physical game board of Fiery Dragons consists of 8 volcano cards(segments), 4 caves, 4 dragons and 16 chit cards. To model this composition, we created aggregation relationships between GameBoard and each of these components with the appropriate multiplicities.

VolcanoCard, Square, ChitCard, LuckCard (extension), Card

The VolcanoCard entity models each of the 8 segments of 3 connected squares in the Fiery Dragons game. This justifies the aggregation relationship between VolcanoCard and Square. As part of our extension, the LuckCard entity represents a set of random cards that can be drawn when the player lands on certain squares. To encapsulate the generic features of the Square, ChitCard and LuckCard entities, we added a generalisation relationship between each of those specific cards with the Card entity.

The main purpose of this design pattern is to support code reuse and extensibility. By defining a generic Card entity, our design can easily be extended to include the LuckCard functionality in future Sprints (Open-Closed Principle)

Animal and DragonPirate

To represent the various animals depicted on the square cards and chit cards, we decided to create Salamander, Spider, BabyDragon and Bat as separate entities. Their generalised entity is the Animal entity, which is the abstract concept that specifies the features of the animal. One such feature is an image attribute which will be rendered in the user interface, and it is specific to each animal.

Both the ChitCard entity and the Square cards that form the Volcano segments should display animals, and therefore an association relationship is formed between Square and Animal, as well as ChitCard and Animal. Each square contains 1 animal, whereas each chit card contains 1,2 or 3 animals. Each chit card can also contain 1 or 2 dragon pirates.

One alternative design choice is to allocate this functionality to the generic Card entity. However, the LuckCard entity does not require the services provided by the Animal entity, and this would result in a trivial implementation. By creating a separate Animal entity, the Animal interface will be implemented by only the entities that require it. (Interface Segregation Principle)

Dragon, HumanPlayer, AIPlayer (extension), Player

The HumanPlayer entity represents the end-users. As part of our extension of the game to implement AI players for the single-player mode, we define the AIPlayer entity, which

functions similarly to human players. We decided to extract this common functionality into a generic Player entity (generalisation relationship) which defines all the common responsibilities such as uncovering chit cards and moving the player's token. These can be implemented as abstract methods that can be applied to any specific player, which supports the Liskov Substitution Principle. We added a 1:2..4 association relationship between GameRound and Player since the game can be played by 2 to 4 players.

4. Basic UI Design

https://www.figma.com/file/Wwzd74lduk5JHXwxuhV8y7/Untitled?type=design&node-id=0%3A1&mode=design&t=jIC1Wbmb3yO3Xro1-1

Starting Page

The above is the homepage for the game, where users can select the game mode.

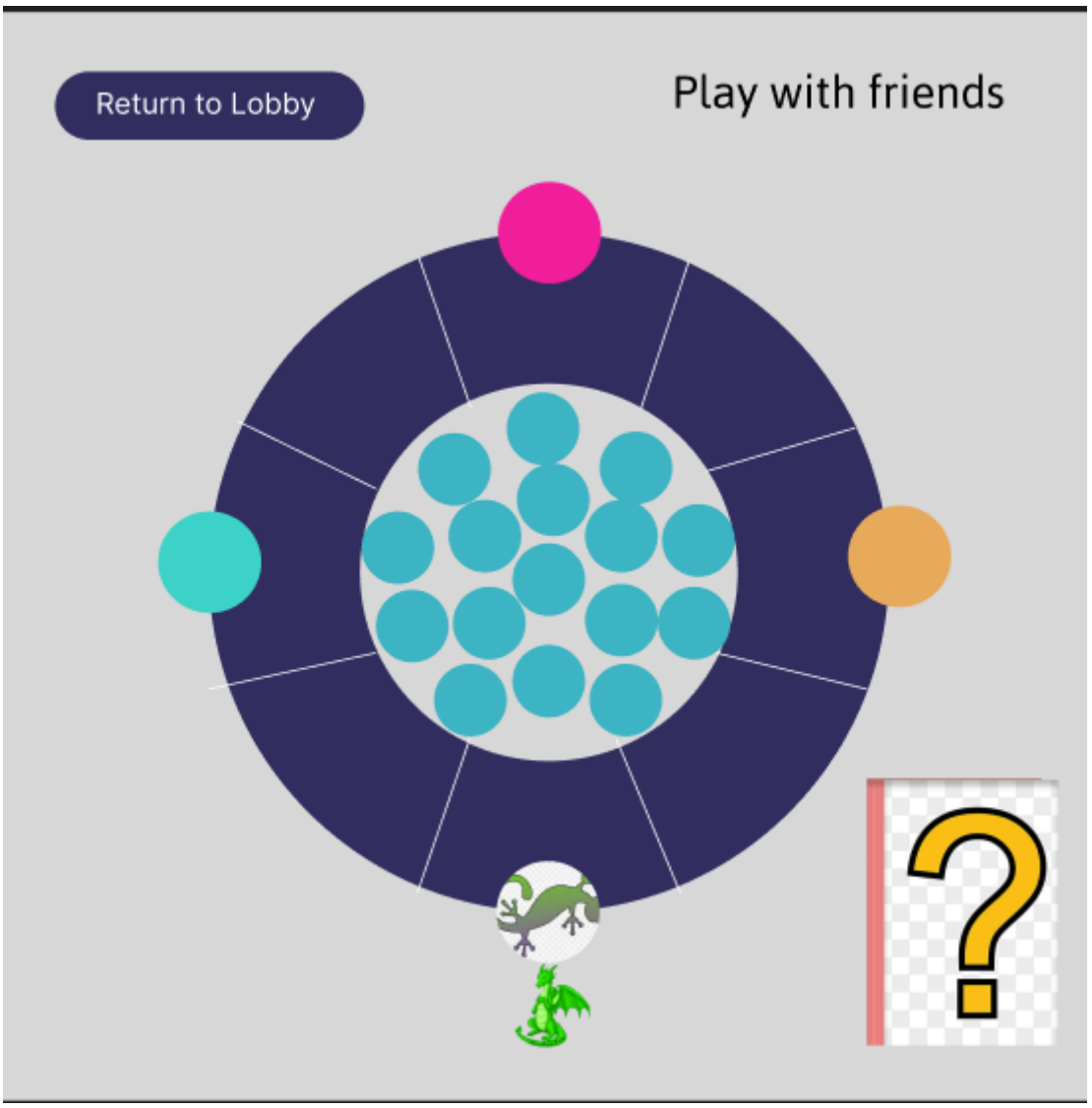
Single Player: This is an *extension* that allows users to play by themselves against 3 other bots

Play With Friends: This is the default game-mode that users are made to play with 1-3 other friends.

Instruction Page

The picture above is the instruction page which demonstrates the basic game rule for the player once they clip either two buttons in the home page. The next button is for them to proceed to the game.

Playing with Friends



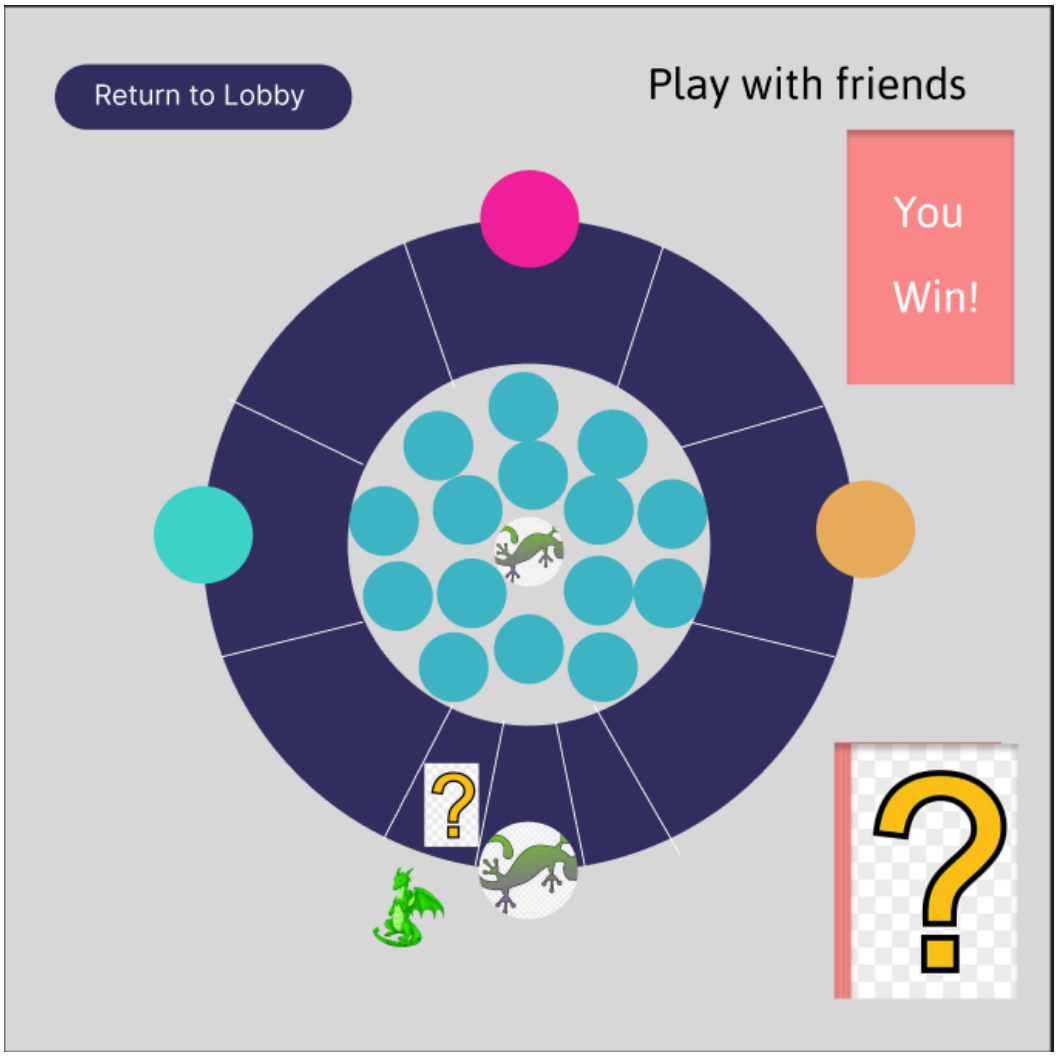
This is the Play With Friends mode where users can play with up to 3 of their friends. Users have to flip **Chit Cards** with animals that match the animal on the square their token is on, in order to progress with the game. The first player who moves their **Token** around the board once back into their **Cave** wins the game.

Green Salamander Cave: It is the cave where the user starts and the user has to flip over a chit card of Salamander type for them to progress

Green Dragon Token: The green dragon is the token which users will use to move around the board to play the game

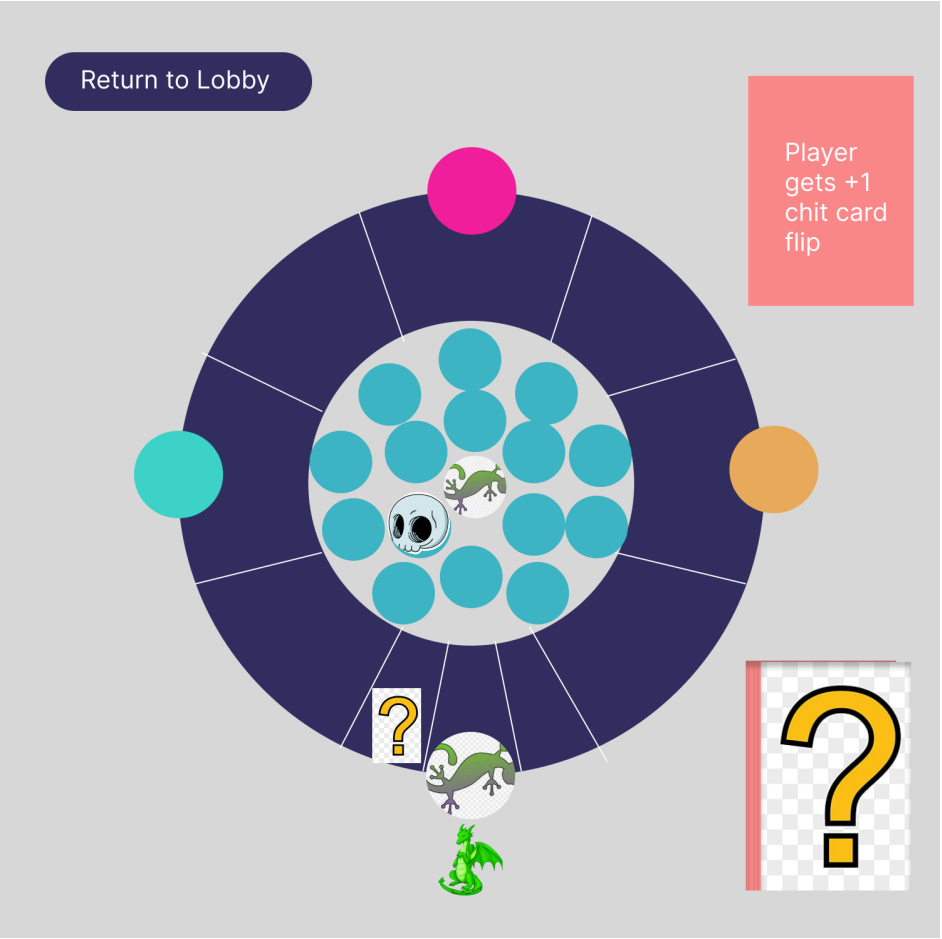
Chit Cards: The chit cards are the 16 cards at the centre of the board where users have to flip over two of the same kind to progress onward the board

Flipping chit cards and moving



Here, the user just flipped over a chit card containing one Salamander. **Since it matches the Salamander on the square the dragon token is on**, the user moves forward one step (In the diagram above, the dragon token has moved forward by one position).

LuckCard (Extension)

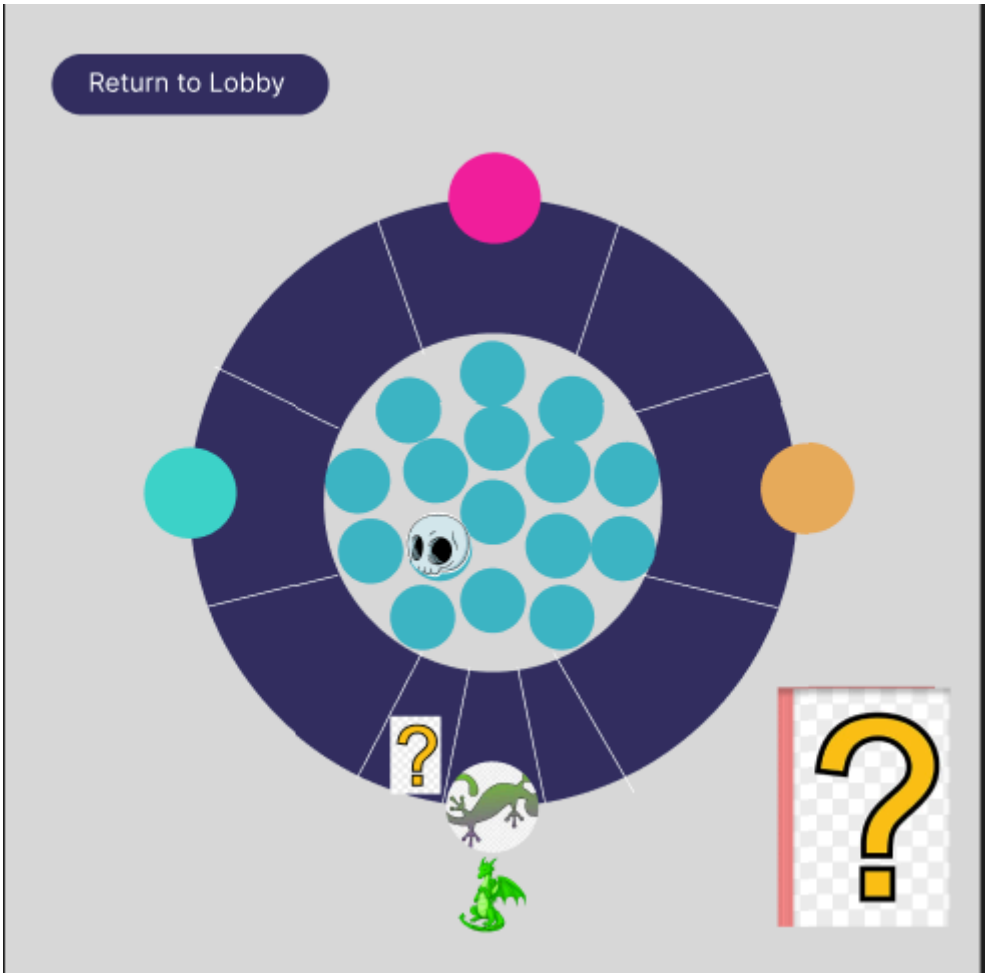


In the previous diagram, the square the player stepped on is also a Luck Zone. ince the square is also a luck zone, the player has to take a **Luck Card** from the top of the deck at the bottom right.

Luck Card: A luck card gives either advantages or disadvantages for the player, the outcome could be up to moving steps forward or winning the game directly, to moving negative steps or not being able to move for round(s).

Win Condition

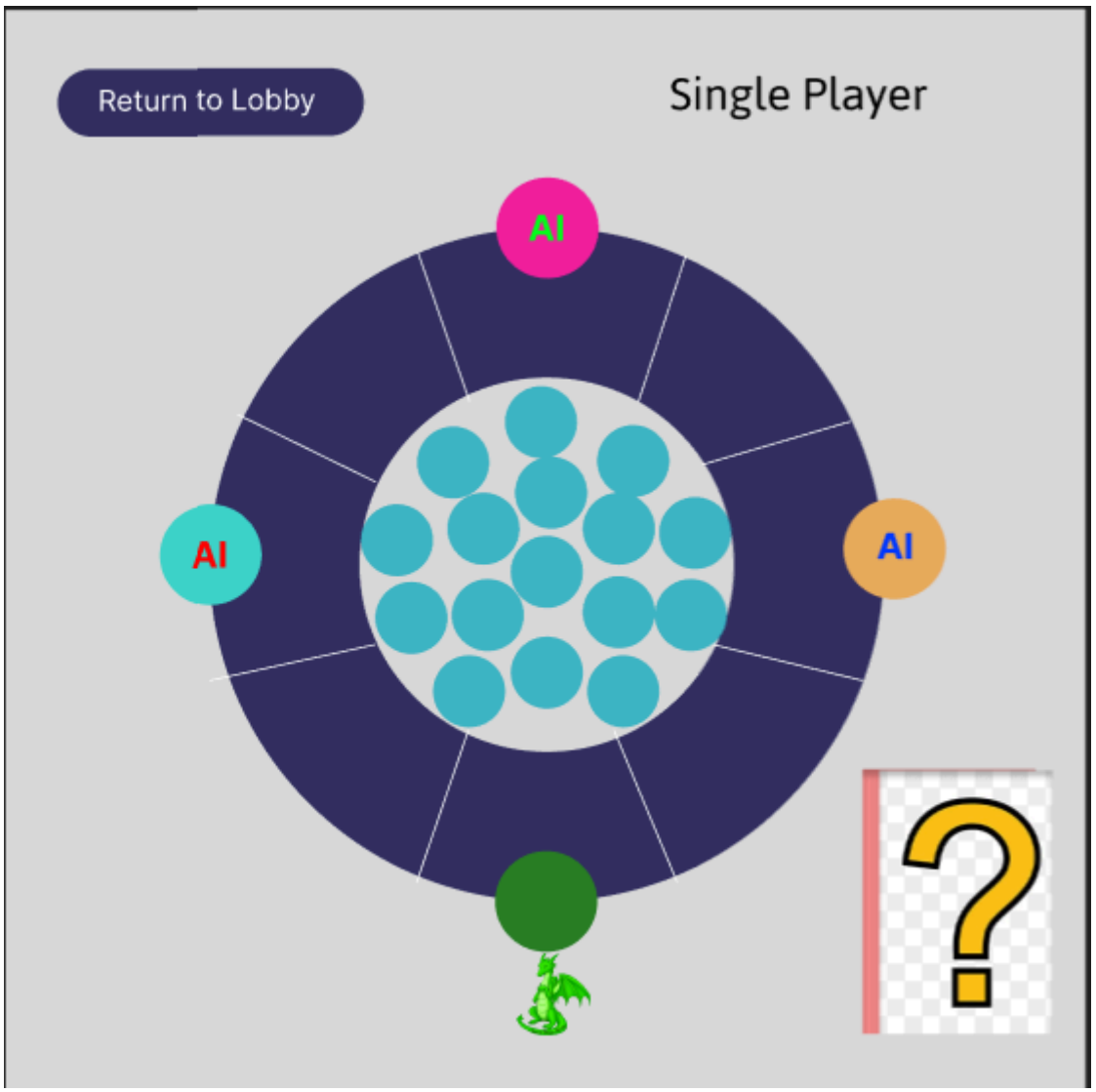
Fifth Page: Playing the Game(2)



If User flips over the **Skull Chit Card** they are required to move backwards a step or however **Skull** there is on the Chit Card

Skull Chit Card: Skull cards are the enemy of the players, when they flip open a skull chit card, they have to move their character token backwards according to the number of skulls on the chit card.

Single-Player Mode (Extension)



Clicking on the Single Player on the Starting Page brings you to the Single Player gamemode where you will be able to play amongst 3 different bots that we implemented as our *extension* of the game. Here, the player starts at the green cave and controls the green dragon token, while the other 3 players are AI-controlled players.

Return to Lobby Button: The return to lobby button will return the user to the starting page where they can again decide to choose the Single Player Gamemode or Play With Friends.

5. Appendix

1. Access to the contribution log can be found in the wiki section of GitLab. Alternatively, the link is also attached below: https://docs.google.com/spreadsheets/d/1HGMoftoWCWHY2sFq8PhkkfQV9qjMpe_db2NZriE1g5k/edit#gid=0

2.