FIT3077 Software Engineering: Architecture and Design
Sprint One Submission

CL_Monday06pm_Team988
Group Report

Group Members:	**Email Address:**
Sheena Quah Xin Yee	squa0013@student.monash.edu
Samantha Oh Jia-Jia	sohh0008@student.monash.edu
Racheal Lim	rlim0050@student.monash.edu
Ng Jeh Guan	jngg0104@student.monash.edu

Table of Contents

1.0 Overview

2.0 Team Information

2.1 Team Name & Photo

Team Name	Team Photo
SATU MALAYSIA	

2.2 Team Members

Member	Strengths	Fun Fact About Me
Racheal Lim rlim0050@student.monash.edu	**Technical Strengths:** ● **Languages:** Java, Python, HTML5, CSS, SQL ● **Tools:** Relational Databases, MySQL, Agile Testing, Adobe Software, Microsoft Office **Professional Strengths:** I am a quick learner and have shown the ability to pick up new skill sets easily with minimal supervision.	I like eating raw durian
Ng Jeh Guan jngg0104@student.monash.edu	**Technical Strengths:** ● **Languages:** Python, R, Java, C++, JavaScript, Haskell, SQL ● **Tools:** Relational Databases, MySQL, Adobe Software, Microsoft Office **Professional Strengths:** I have a strong basic knowledge of several	I like reading

	programming languages and tend to be able to adapt to new programming languages quickly.	
Samantha Oh Jia-Jia sohh0008@student.monash.edu	**Technical Strengths:** ● **Languages**: Python, Java, HTML, CSS, Javascript, Haskel ● **Tools**: Relational Databases, Microsoft Office, Adobe Software, MySQL **Professional Strengths:** ● I have a strong basis in programming and always think outside the box.	I LOVE CATS
Sheena Quah Xin Yee squa0013@student.monash.edu	● **Technical Strengths:** Languages: Python, Java, HTML5, CSS, Javascript ● **Tools:** Relational Databases, Microsoft Office, Adobe Software, MySQL **Professional Strengths:** I am a creative person who tends to come up with good ideas. I also learn new skills rather quickly.	(I LOVE CATS^2) AND DOGS

2.3 Team Schedule

2.3.1 Meeting & Work Schedule

Our team holds weekly meetings **every Monday** to discuss our progress, address any issues that have arisen, and plan for the week ahead. These meetings are crucial for keeping everyone aligned and informed about the project's status and any challenges we may be facing.

Date, Time and Mode	**Members Present (list those present)**	**Discussions**
15/03/2024, 19:00-20:00, physical meeting	● Sheena Quah Xin Yee ● Samantha Oh Jia-Jia ● Racheal Lim ● Ng Jeh Guan	● Set up group expectations ● Get together as a group and Ice-breaking
18/03/2024 19:00-20:00,	● Sheena Quah Xin Yee ● Samantha Oh Jia-Jia	● Discussion on user stories ● Set up report structure

physical meeting	● Racheal Lim ● Ng Jeh Guan	● Prototyping
25/3/2024 14.30-19.00 physical meeting	● Sheena Quah Xin Yee ● Samantha Oh Jia-Jia ● Racheal Lim ● Ng Jeh Guan	● Resolves issues and challenges ● Progress Update ● Prototype Finalizing ● Domain Model

2.3.2 Workload Distribution and Management

Task allocation will be determined by each team member's competence, availability, and workload balance. All tasks will be equally divided among team members for each section provided in the report.
Regular Check-Ins: Team members will communicate with one another on a regular basis to ensure progress and give assistance as required.
Communication: Clear and open communication will be maintained to address any workload difficulties or necessary changes. We will utilize a discord server as our medium of communication to keep track of projects and progress.

2.4 Technology Stack and Justification

Our team has chosen to utilize **Java as the main programming language** for the Fiery Dragon game project. This selection is in line with the present knowledge of our team because all of the members have finished an introductory unit on Java OOP, giving everyone a solid foundation in the language. We can create the game more quickly by making use of our prior Java expertise and concentrating more on the mechanics and design of the game than on picking up a new programming language.

Furthermore, Java is a flexible language with a large selection of libraries and tools appropriate for game creation. This contains the JavaFX library, which we want to utilize to develop the graphical user interface (GUI) for the game. We can create engaging and aesthetically pleasing user interfaces for the game thanks to JavaFX's smooth integration with Java.

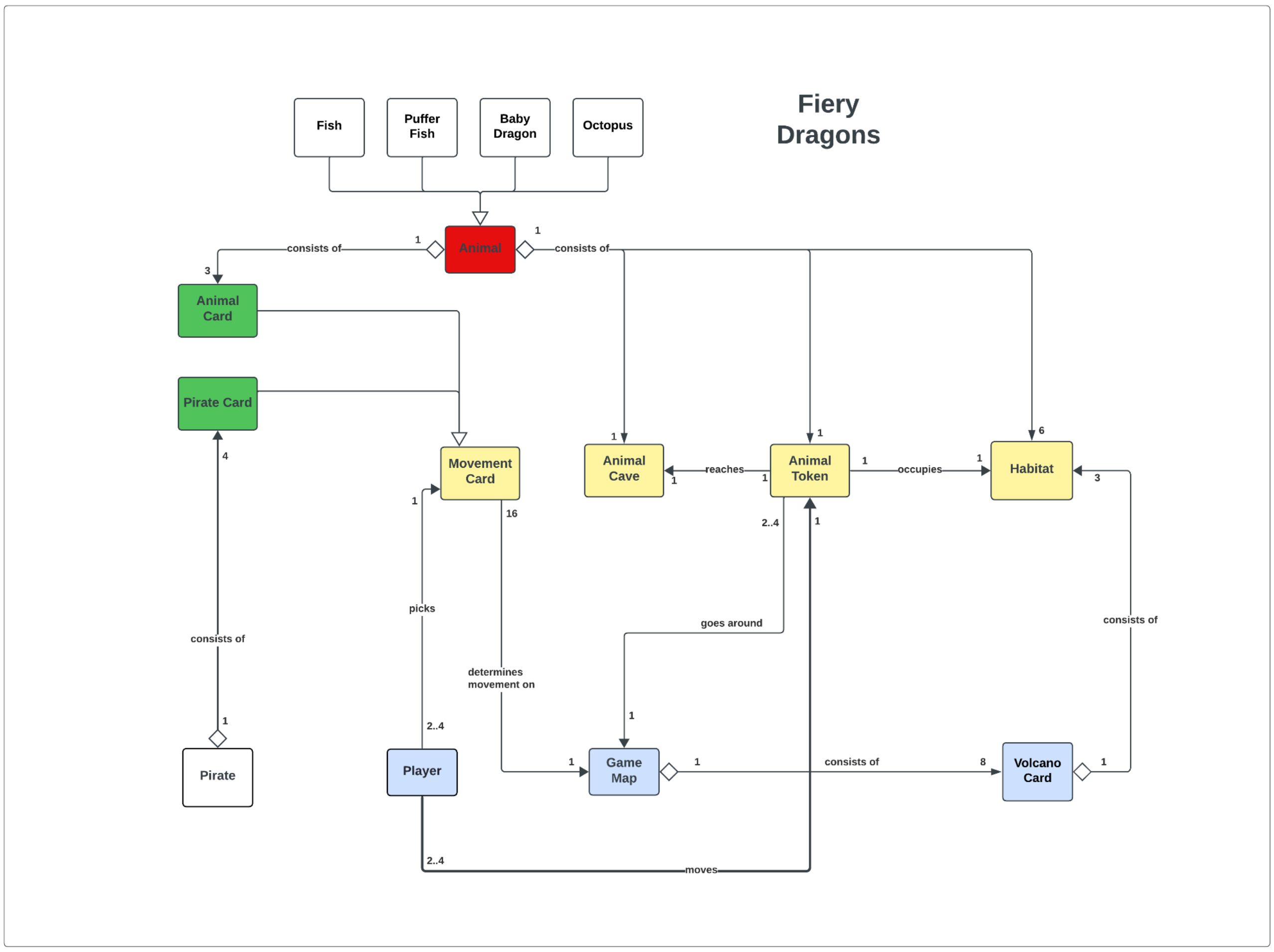
Although our team is confident in our abilities to work with Java, we expect that for more difficult issues relating to game creation, we will require assistance from the teaching team. This might include some instructions and guidance on the game implementation.

3.0 User Stories

No.	User Stories
1	As a player, I want to uncover the animal cards one by one, so that I can memorize all the card locations.
2	As a player, I want to end my turn by covering up all the animal cards, allowing the next player to take their turn, so that the game flows smoothly and each player has a fair opportunity to play.
3	As a player, I want to uncover an animal card and move my animal forward based on the number of animals shown on the card, if it matches the animal on the square where my animal stands, so that I can progress around the volcano.
4	As a player, I want to move my animal with the exact number of moves required to reach my cave, so that I can follow the rules of the game and maintain a fair gameplay experience.
5	As a player, I want to be able to send an opponent back to their cave upon landing on a tile occupied by them, so that I can take advantage of being in the lead when the opponent has to start over again. (extension)
6	As a player, I want to place my uncovered animal cards back in their original positions after my turn, so that I can know which animal cards benefit me by memorizing their positions.
7	As a player, I want to have instructions on how to play the game so that I can understand the objectives and rules.
8	As a player, I want to be able to have an extra turn after successfully making a forward move, so that I can quickly advance back to my cave and win the game.
9	As a developer, I want to put all the animals in their caves before the game starts, so that I can ensure that each player's animal is displayed properly in the right position.
10	As a player, I want to be able to view a clear and presentable game user interface, so that I can observe the board change on each turn of the game.
11	As a developer, I want to randomly select a player to move first at the start of a new game, so that all players can take turns to make the first move, balancing the game.
12	As a developer, I want to limit players to only be able to pick one card per turn, so that each player has a fair opportunity to play.
13	As a developer, I want to test the game before publishing it to ensure the gameplay is smooth.
14	As a player, I want to have different animal characters to choose from, so that I can customize my gaming experience.

15	As a developer, I want to limit the total number of habitats of each type of animal on the game board to be 6, so that the sum of the habitats of all types will be 4*6 = **24** which is a fair game for the 4 players.
16	As a developer, I want to limit players who have an extra turn to leave their previously selected animal card uncovered, so that whenever a player has an extra turn, the game will not be exploited.
17	As a developer, I want to limit players to only move when the animal cards they picked match the animal on the tile they are currently on, so that players will have to memorize the location of all types of animal cards.
18	As a developer, I want to create tiles with different animals, so that whenever a player lands on a tile that is different from the previous round, a different animal card is required for them to move forward.
19	As a developer, I want to limit the amount of moves a player can move to at most 3 moves per round, so that the game can go on for a reasonable amount of time.
20	As a developer, I want to limit player movement such that when an animal pirate card has been selected and the player moves backwards accordingly, the player does not get an extra turn, so that it prevents the player from getting another chance to move forward.

4.0 Domain Model



Several design decisions were made when building the domain model for "Fiery Dragons" to represent both the game's rules and the underlying structure, with the goal of making future expansions and adjustments easier. Here is a breakdown of the relationships and justification.

Animal - Inheritance Hierarchy
The usage of an Animal superclass makes it straightforward to introduce more sorts of animals in the future. By inheriting from Animal, new creatures may be produced with little code duplication, ensuring that they have all of the essential features without having to manually add them to each habitat or cave.

The Animal class has a one-to-many relationship with the Animal Card. Each Animal type can have multiple cards associated with it, following the game's rule where each animal type has 3 movement cards (go forward 1, 2, or 3 steps).

This relationship between Animal and animal token is slightly more abstract in the game context, but it can be seen as one-to-one. One type of Animal can correspond to one Animal Tokens on the game map, depending on the number of players and tokens in play.

Animal Tokens and Caves
Linking each Animal Token directly to an Animal and then to an Animal Cave implies that each animal type has a distinct destination, similar to the game rule that each dragon token reaches its appropriate animal cave. This design creates a clear path for each player's movement throughout the game.

Each Animal Token has an association to one Animal Cave, creating a one-to-one relationship that conforms to the rule that each dragon token has a unique destination cave on the game map.

Habitats as Collections
Each habitat contains collections of animals, in accordance with the rule that the game map is built up of several sets of volcano cards, each represented by an animal-containing habitat. The habitats are structured as groups of volcano cards, making it simple to comprehend how the game world is put together and how players may interact with it.

Game Map Composition
Based on our model, the Game Map is made up of Volcano Cards, which are made up of Habitats, which in turn include Animals. This stacked composition illustrates the tiered nature of the game board and how players move around it.

The Game Map has a one-to-many relationship with the Volcano Cards. The gaming map is made up of 8 sets of volcano cards. Because a volcano card consists of several habitats, it has a one-to-many connection with them. The relationship between habitat and animals is one-to-many. Each habitat has 3 different animals (3 sets of three animals, according to the game regulations).

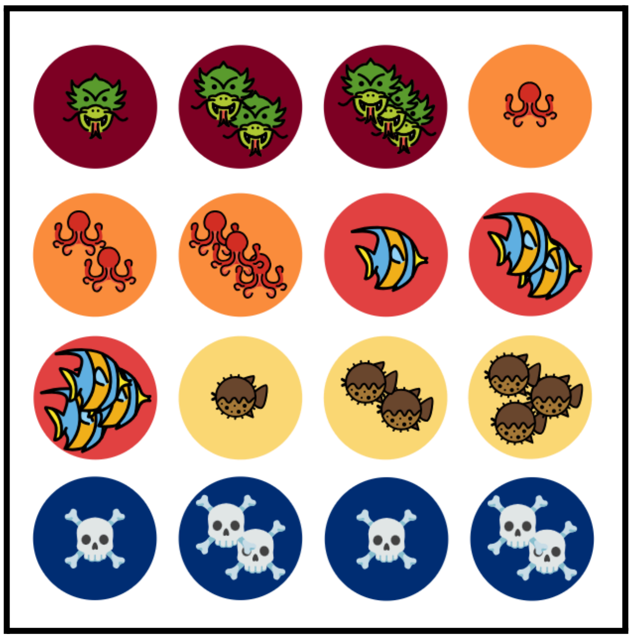
Player Movement with Cards
The domain model distinguishes between Animal Cards and Pirate Cards based on their opposing impacts on movement. This division is rational and consistent with the game's rules, which state that animal cards always cause forward movement while pirate cards cause backwards movement. The model's clarity makes it easier to grasp the game's flow.

The model uses numbers to show how many of each sort of card or token are linked to other elements. For example, the Player selects Movement Cards, which are divided into Animal and Pirate Cards. These multiplicity indications assist in comprehending the amounts involved in gaming, such as the number of steps to take.

Multiple Players have a many-to-one relationship with Movement Cards, as they can only pick one animal card at a time throughout the game.

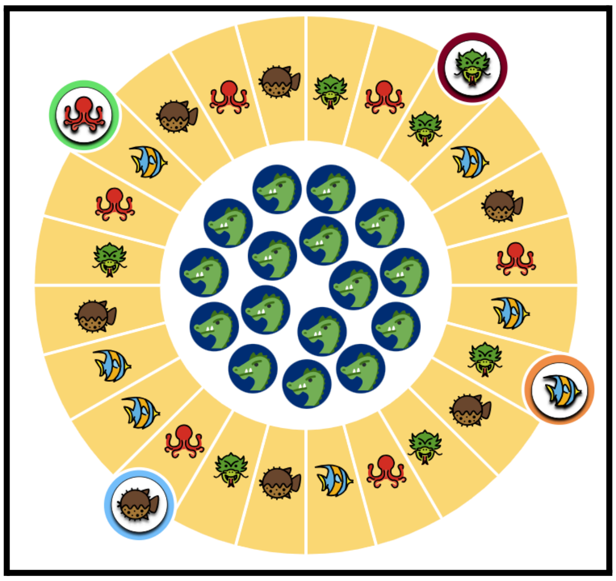
5.0 Basic UI design

5.0 Uncovering Dragon Cards of Various Types



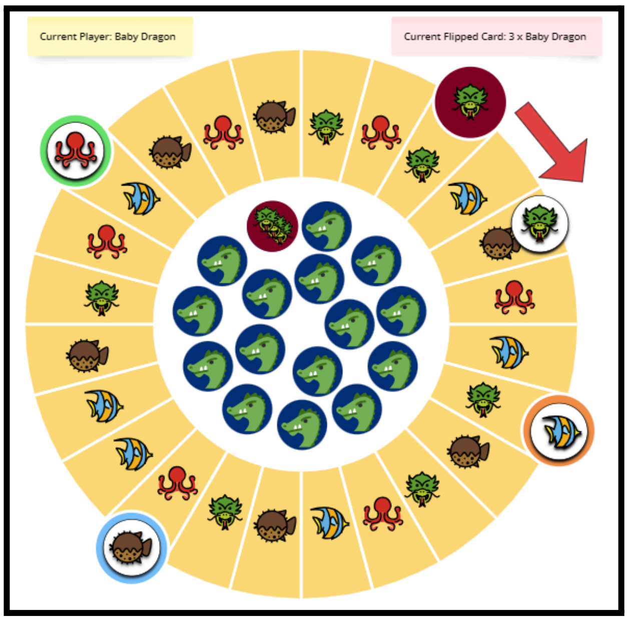
5.1 Initial Game Setup

This is the **initial game setup** that shows all animal tokens in their respective cave. 16 covered animal cards are placed in the middle of the game map with 24 habitat cards that form the structure of the game map.

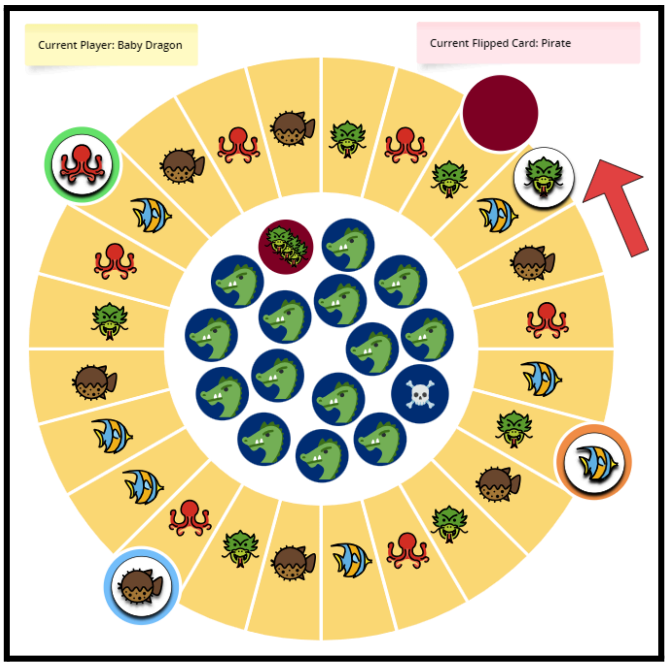


5.2 Standard Game Rules

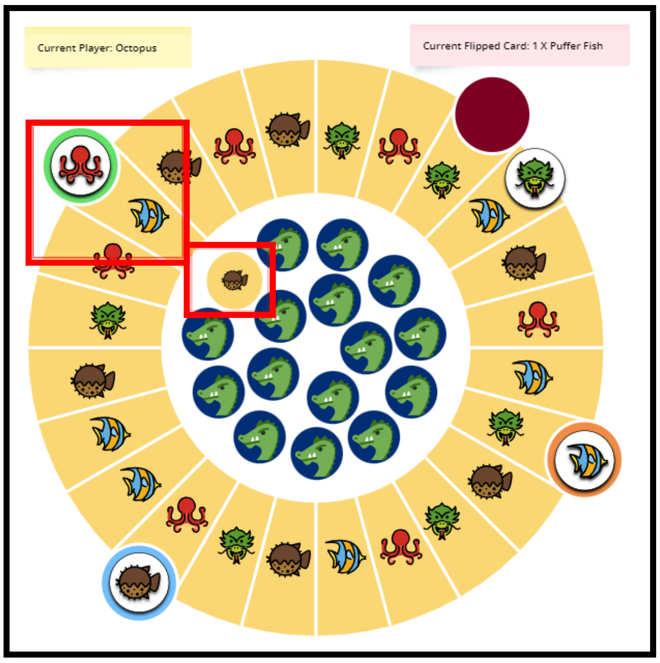
It is currently "Baby Dragon's Round," and the respective player has the chance to navigate their animal tokens across a volcanic landscape. During their turn, players flip over an animal card. The animal card reveals three Baby Dragon icons, the player moves the baby dragon token three steps clockwise on the game map. Additionally, the player can continue flipping over cards, potentially uncovering more Baby Dragon icons to continue moving their token.


Current Player: Baby Dragon
Current Flipped Card: 3 x Baby Dragon

As the figure shown below, when a player flips over a card revealing a single **Pirate** icon, their dragon token must move one step backwards (anticlockwise) on the game map. Unlike other flips, players cannot continue to flip additional cards after revealing a Pirate. Furthermore, all cards that have been flipped during this turn are turned back over, resetting the game state.

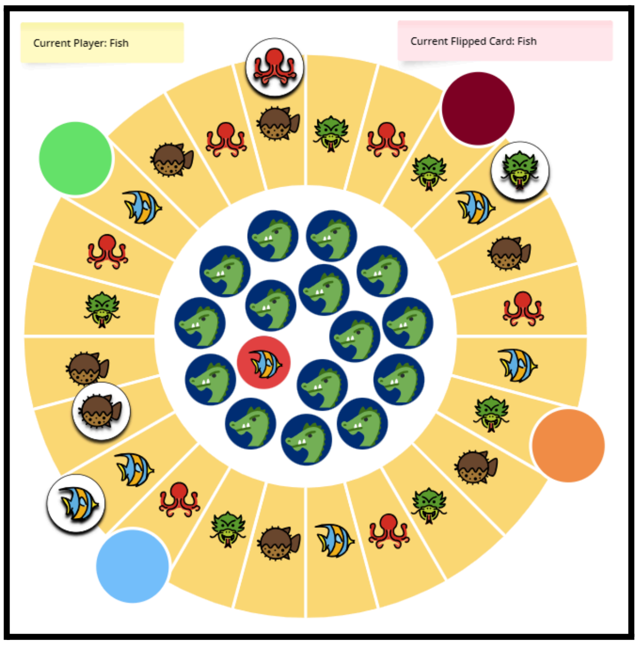


Moving on to the next player (Octopus), the octopus player turns over a card that reveals a single Puffer Fish card, the Octopus cannot move since it does not match the character (Fish) on the current map. Like meeting a Pirate, all flipped cards are turned back over, restoring the game state. The next player takes his or her turn, and the gameplay continues.

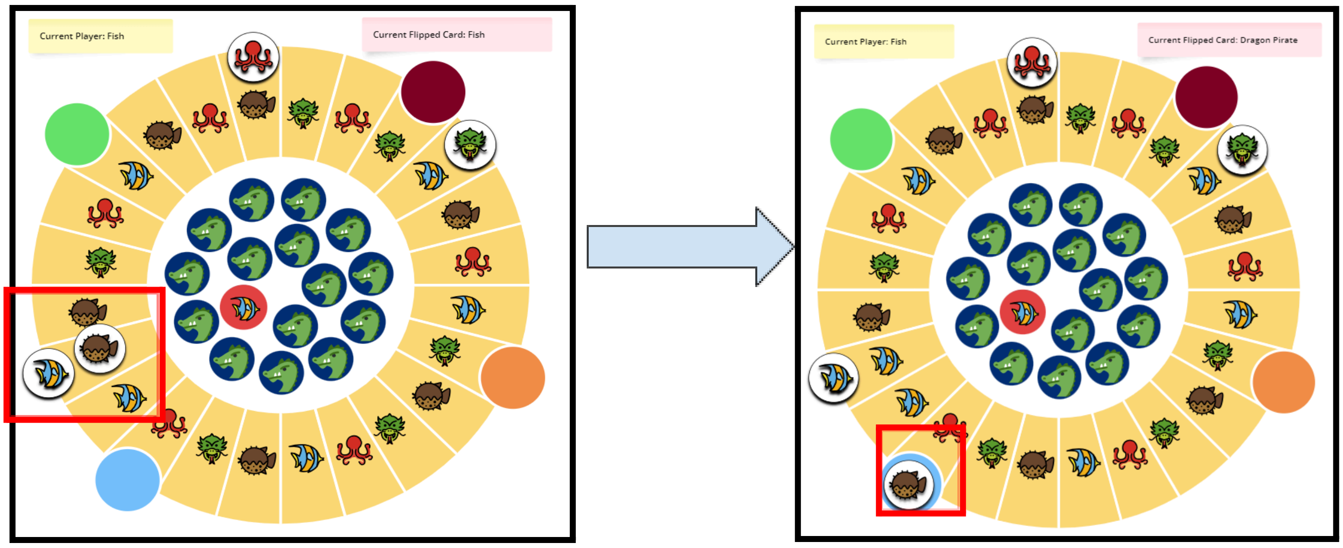


5.3 Extended Game Rules

In "Fish's Round," when the current player, controlling the Fish, flips over a card revealing a single Fish icon, the Fish moves one step forward on the game map. This movement occurs because the Fish matches with its current map's animal.

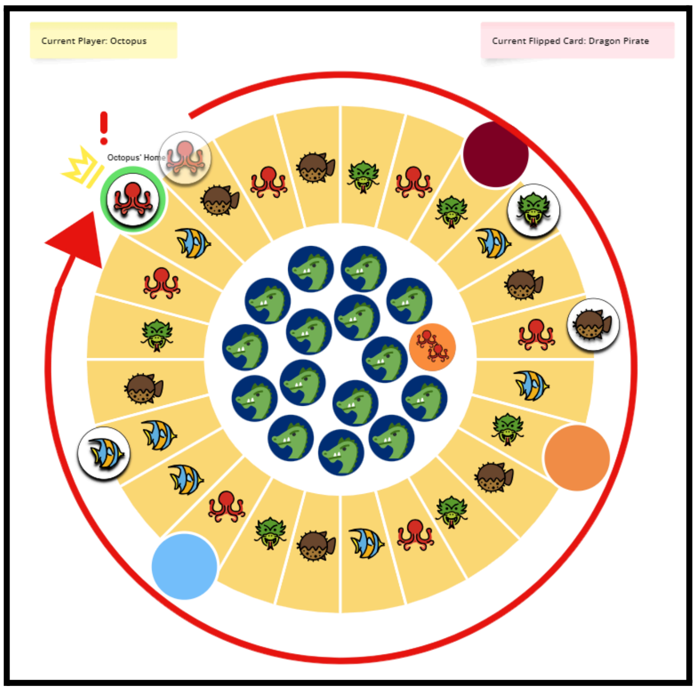


After moving one step forward, the fish token lands on the animal habitat card occupied by a Puffer Fish token, the Puffer Fish token is then sent back to its cave, restarting the entire round again.


Current Player: Fish
Current Flipped Card: Fish
Current Player: Fish
Current Flipped Card: Dragon Pirate

5.4 Winning Situation

The player who successfully navigates their animal token to their respective animal cave first wins the game. In the figure below, Octopus successfully completed one cycle of the entire game map and got back to its cave first. Hence, the player for the **Octopus token is the winner for this round**.


Current Player: Octopus
Current Flipped Card: Dragon Pirate
Octopus' Home