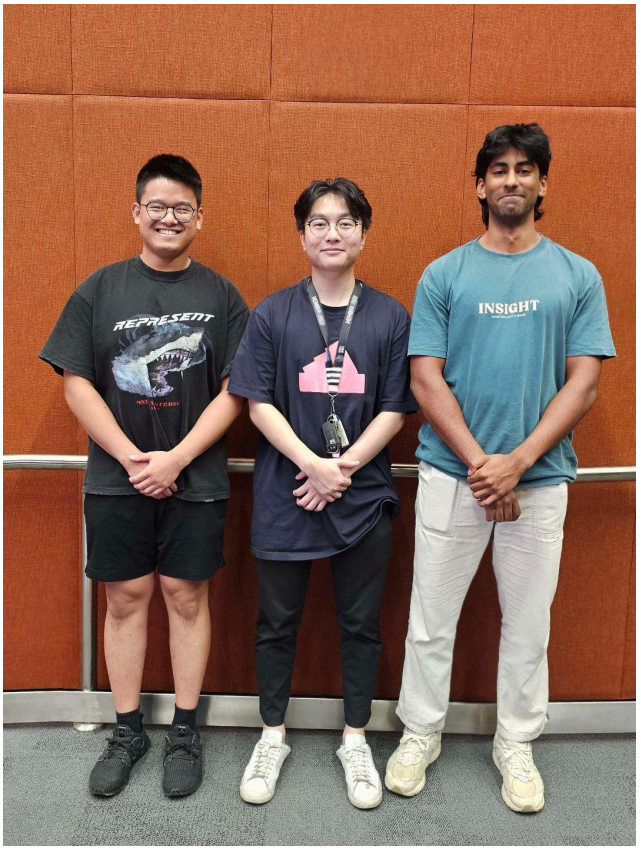
# FIT3077 Assignment 1 - Specifications

## Team 149 - Object Oriented Dragons



# Table of Contents

## Team Information

| Name | Contact | Technical and Professional Skills | Fun Fact |
|---|---|---|---|
| Zilei Chen | zche0160@student.monash.edu<br>0481 189 932 | Java, Python, Leadership | I have 6 cats |
| Jeffrey Yan | jyan0160@student.monash.edu<br>0432 818 393 | Java, Python, Testing and Debugging | I own 4 keyboards for fun |
| Krishna Pillaai Manogaran | kman0030@student.monash.edu<br>0416 763 341 | Java, Python, Communication | I have been skydiving |

**Team Schedule**

Meetings will be scheduled for 6-8pm on Mondays in person, and 2-4pm on Wednesdays over Discord. Additional meetings will be set accordingly if necessary. In these meetings we will delegate tasks to have attempted or completed before the next meeting so that we can analyse progress, assess areas we may be having difficulty in and redistribute work. This will also incorporate standup meetings to inform everyone what has been done. It will be expected for all team members to attend all team meetings, unless notice is given for unavailability.

Tasks will be documented in a backlog on discord and allocated during the Sprint Planning meeting held at the beginning of each sprint. Priority for tasks will be given to team members who show an interest in a certain task, or team members who may have previous experience in the required area. The remaining tasks in the backlog will be distributed amongst all team members accordingly, to ensure there is an even spread of work. In the event that a team member has trouble with work, they are expected to communicate their difficulties to the rest of the team via Discord, where assistance can be provided and problems further discussed at the next meeting.

**Technology Stack and Justification**

We intend to use Java as our programming language. All team members have experience working with both Java and Python, it came down to what each language offered in terms of building a turn based game. Java was the chosen language due to our familiarity with the language and JavaFX is the preferred API of choice since it allows us to create a customisable window which we are able to put in the game and various animations. The skills required to operate and code with JavaFX is minimal since they are simply Java classes that have been written for us. Other options were considered including Swing, which is an older generation of GUI API for Java, although inbuilt. Some reasons for discarding Swing for JavaFX were the simplicity of using multimedia in JavaFX, along with integrating with HTML and CSS. Additionally, as JavaFX is a newer technology, there would be more up-to-date information and relevancy, and we preferred learning newer technologies as we are more likely to use them in the future.

## User Stories

The following user stories will be used to represent the base Fiery Dragons game. Every feature necessary for the base game to be playable will be documented here, including setup, rules and gameplay requirements.

**Setup:**

1. As a player, I want to be able to select a number of players between 2-4, so that I can accommodate the game to how many friends are with me.
2. As a player, I would like to be able to restart a game once it has been completed, so that I can continue playing without restarting the application.
3. As a player, I want the MapPieces arranged in a different configuration, so that each game I play isn't repetitive and stale.
4. As a developer, I want the game to look visually appealing, so that players are likely to continue playing the game.

**Gameplay:**

5. As a player, I want to be able to know what cave I started at, so I know what type of dragon card I need to pick in order to move.
6. As a player, I want to be able to see exactly where all players are on the board, so I know what moves I need to make in order to win.
7. As a player, I want to be able to see and select all Dragon Cards so that I can advance on the board.
8. As a player, I want to be able to pick a Dragon Card that was revealed before by another player, so that I can ensure that the relevant Dragon Card can help me progress on the board.
9. As a player, I want to see the animals on each tile of the MapPiece, so that I know what Dragon Card I need to pick up next.
10. As a player, I want to be able to see whose turn it is, so that I know who should be operating the computer to play.
11. As a Dragon (Player's token), I want to be able to advance to any tiles on the Board with the exception of my opponent's dragon caves, so that I can progress in the game.

**Rules:**

12. As a developer, I want to ensure that the MapPieces with Caves are separated by at least one normal MapPiece, so that the game can be played as per the rulebook.
13. As a developer, I want a Dragon to be able to move forward if the player selects an appropriate Dragon Card and they are not blocked by another Dragon, so that they are rewarded for their good memory.
14. As a developer, I want a Dragon to be moved backwards if the player selects a Dragon Pirate Card and they are not blocked by another Dragon or still in its cave, so that they are punished for their bad memory.
15. As a developer, I want a Dragon to not be moved if the player doesn't select an appropriate Dragon Card that isn't a Dragon Pirate Card, so that they are punished for their bad memory.
16. As a developer, I want the value of the Dragon Card to be hidden before the next turn, so that other players must rely on memory to select a card they want.

17. As a developer, I want the game to be played in turns until a Dragon completes the circuit, so that it is fair for all players.
18. As a developer, I want the game to end only when a Dragon reaches back its cave in the exact number of moves necessary, so that memory is required to win the game.
19. As a developer, I want a Dragon to not move if the player selects an incorrect dragon card and would pass by their cave as a result of the card, so that the player can have another chance at winning.
20. As a developer, I want to ensure that there is only one dragon per tile on a MapPiece, so that the game can be played as per the rulebook.

There are currently three main extensions that we intend on implementing to the basic Fiery Dragons game, which are as follows.

**Selectable values for the number of MapPieces and tiles per MapPiece:**

1. As a user, I want to be able to choose how many MapPieces are used to create the ring, so that I can have shorter or longer games.
2. As a user, I want to be able to choose how many tiles there are on each MapPiece, so that I can have shorter or longer games.

**Sound effects for actions, such as when a piece moves across the board, flipping over a dragon card, and potentially a positive or negative sound depending on the result of the flipped card:**

3. As a user, I want to be able to hear practical sounds for actions such as moving a piece on the board or flipping over a dragon card, so that the game feels more like a board game and creates a better immersive experience.

## Domain Model

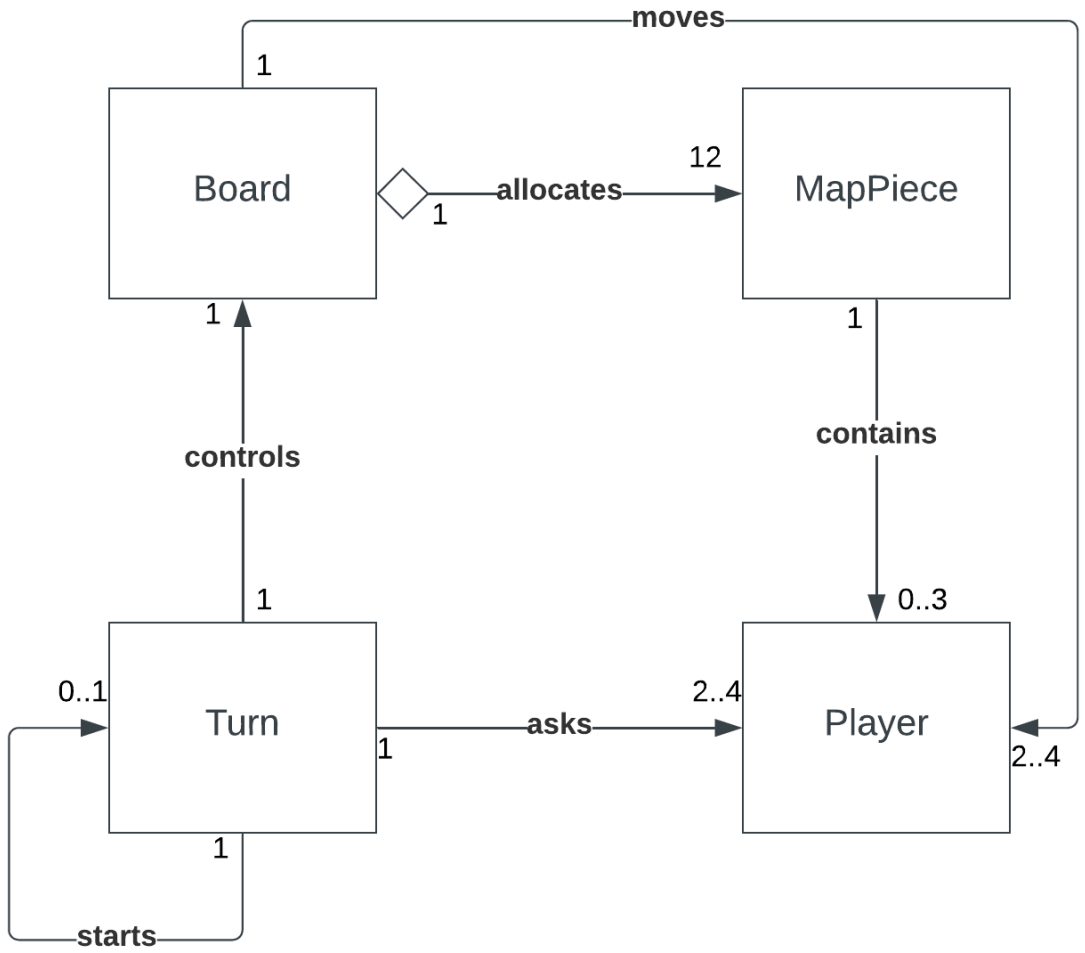



*Figure 1. Domain model for the base game of 'Fiery Dragons'*

To derive a domain model for the game of 'Fiery Dragons', we first need to consider what 'responsibilities' the system needs to complete:

1. **Initialise the board state and start the game.**
2. **Be able to vary the starting board state between games.**
3. **Be able to move the player/dragon token.**
4. **Let the player choose and flip a Dragon Card.**
5. **Check for win condition after every move (player moves back to Cave).**
6. **Detect which player's turn it is and perform their turn.**

The four entities that can take on each of the outlined responsibilities are as follows:

**Board**:

This entity serves as the 'table' of our game, where all game objects are 'placed on', i.e. the container for all Volcano Cards, Cave Cards and Dragon Cards. This entity will take on the following responsibilities:

3. **Be able to move the player/dragon token.**
2. **Be able to vary the starting board state between games.**

Therefore, it has an association with Player, our Dragon Token entity as it needs to be able to move the player. An aggregation exists between Board and MapPiece, since Board contains a group of MapPieces, to

form a more complex 'board state'. Note that there are 12 MapPieces, 8 for Volcano Cards and 4 for Cave Cards.

**MapPiece**:
This entity represents both Volcano Cards and Cave Cards. In the base game, each MapPiece can hold up to 3 Dragon Tokens, depending on whether it is a Volcano or a Cave. This entity is responsible for knowing whether or not a player has returned back to their cave, therefore it is responsible for:

5. **Check for win condition after every move (player moves back to Cave).**

MapPiece has an association with Player, as each MapPiece can contain a Dragon Token when they have moved to a tile on a MapPiece. Note that for the base game, a Volcano Card has 3 tiles, therefore it can contain up to 3 players.

**Player**:
In this domain model, the player entity represents the Dragon Token, as opposed to the physical player. However, it is still responsible for eliciting the response of the player, i.e., finding out which Dragon Card the player chooses to flip. Therefore, its has the responsibility:

4. **Let the player choose and flip a Dragon Card.**

**Turn**:
This entity represents the abstract concept of taking a turn in a physical game of 'Fiery Dragons', and is crucial for actually allowing the game to progress. It will need to find out which player's turn it is, what card the player chooses to flip, then be able to tell the board what to do after finding out this information. After a turn finishes it then needs to start the next turn. As such, it is responsible for the following responsibilities:

1. **Initialise the board state and start the game.**
6. **Detect which player's turn it is and perform their turn.**

Turn has an association with Player, since Turn needs to know which Dragon Token needs to choose a Dragon Card and move next. Turn has an association with Board as it needs to be able to create the initial game board to start the first Turn. Turn also needs to be able to refer back to itself as a new Turn can only begin after an old Turn has finished, so Turn will need to check against itself to see that a Turn has finished.

This method of introducing as few entities as possible means that during implementation, we can reduce the number of Class objects in the overarching design and reduce the complexity of the design. Initially, 8 total entities were created from each component of the game, and after assigning the proposed responsibilities to each entity, a final design was reached (Figure 1) and the following entities were omitted:

1. VolcanoPiece
2. CavePiece
3. DragonToken
4. DragonCard

All these 'inherent' and 'crucial' parts of the game can be represented in ways that don't require them to be entities, and as such we can maintain a manageable level of complexity in the design.

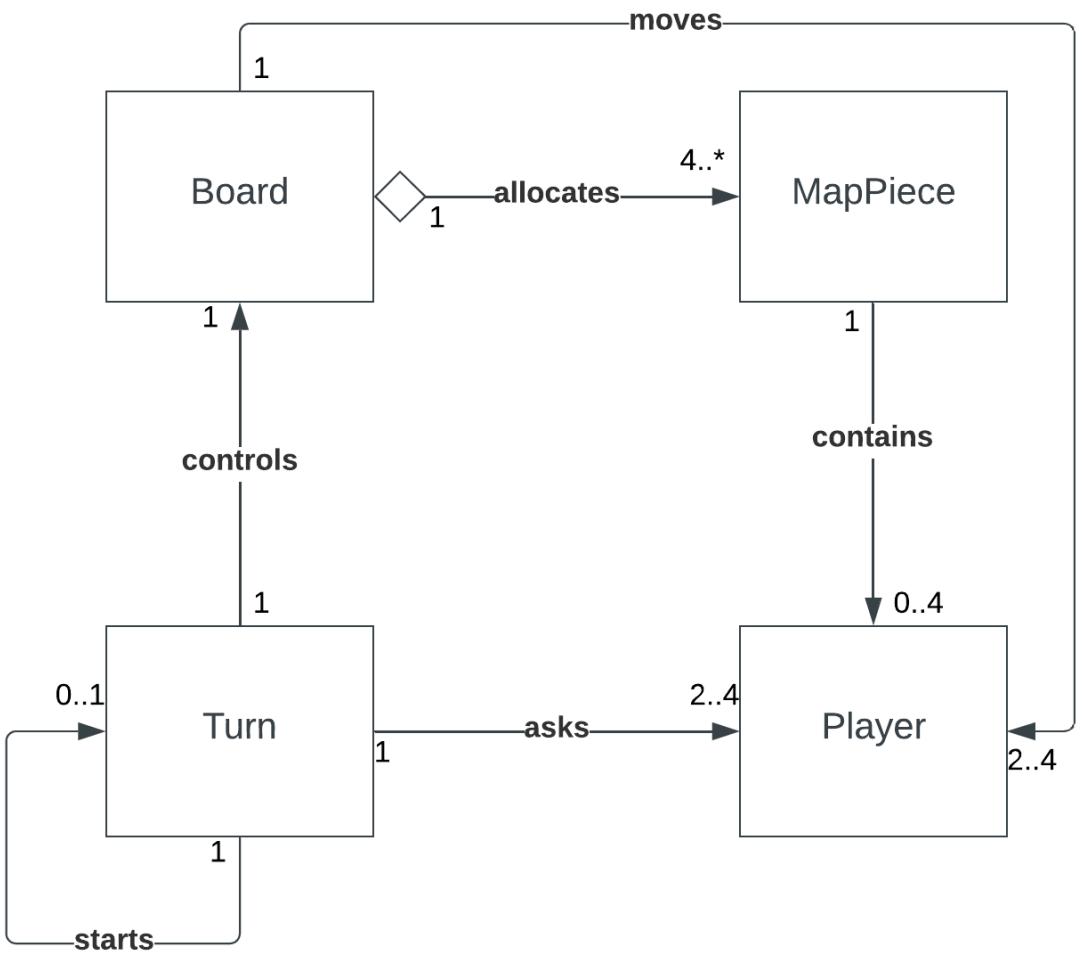


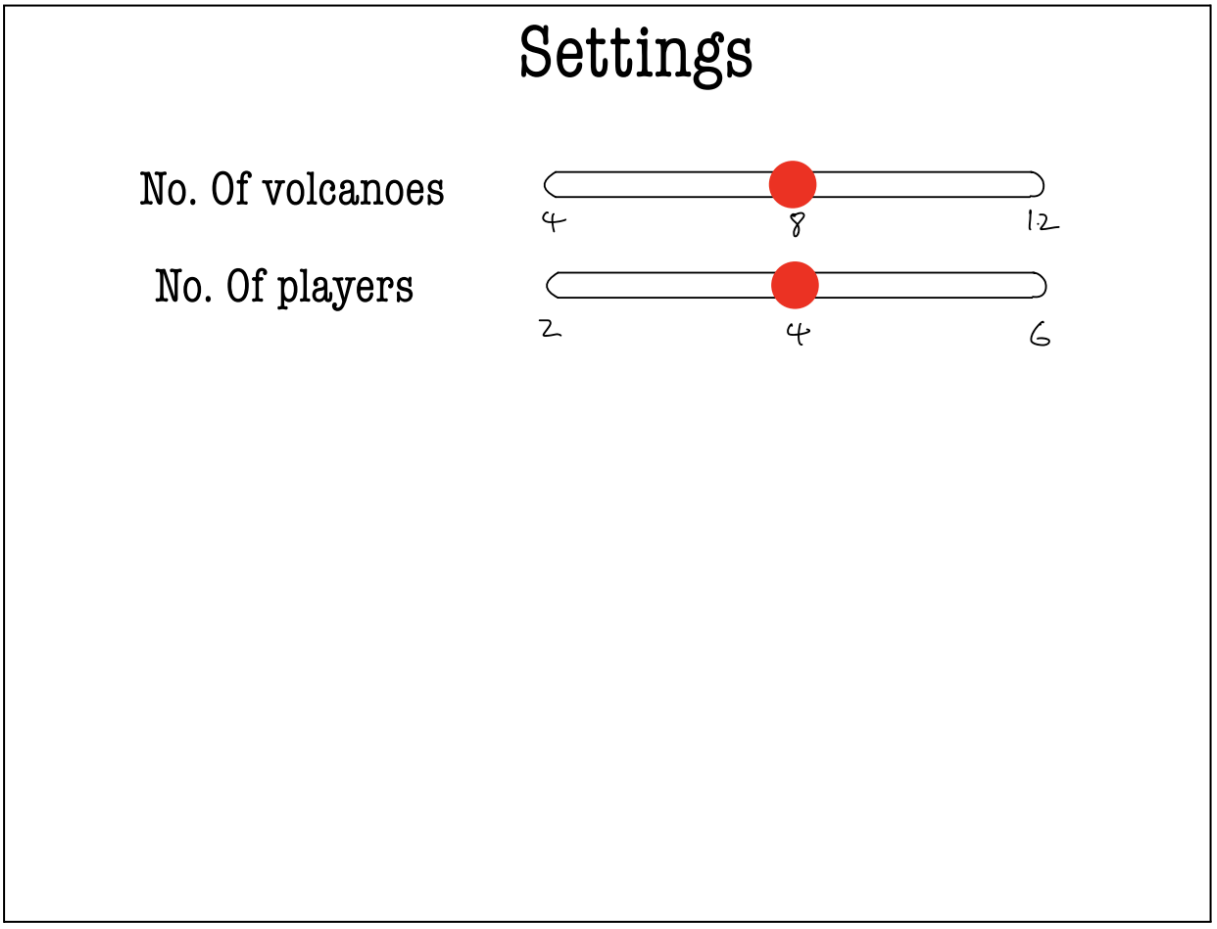

*Figure 2. Domain model for the extensible game of 'Fiery Dragons'*

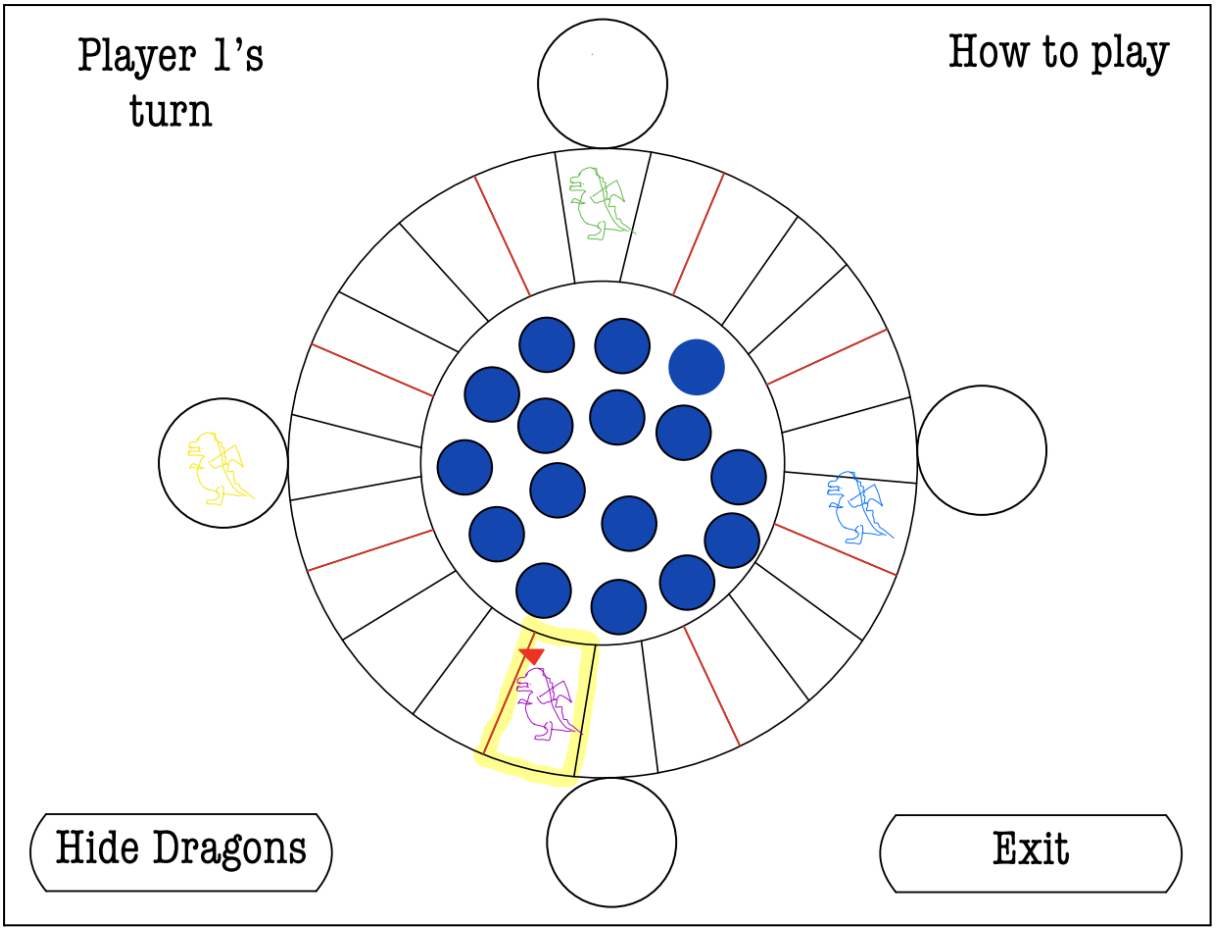
The above domain model shows the changes that need to be made in order to cover the possible extensions where:

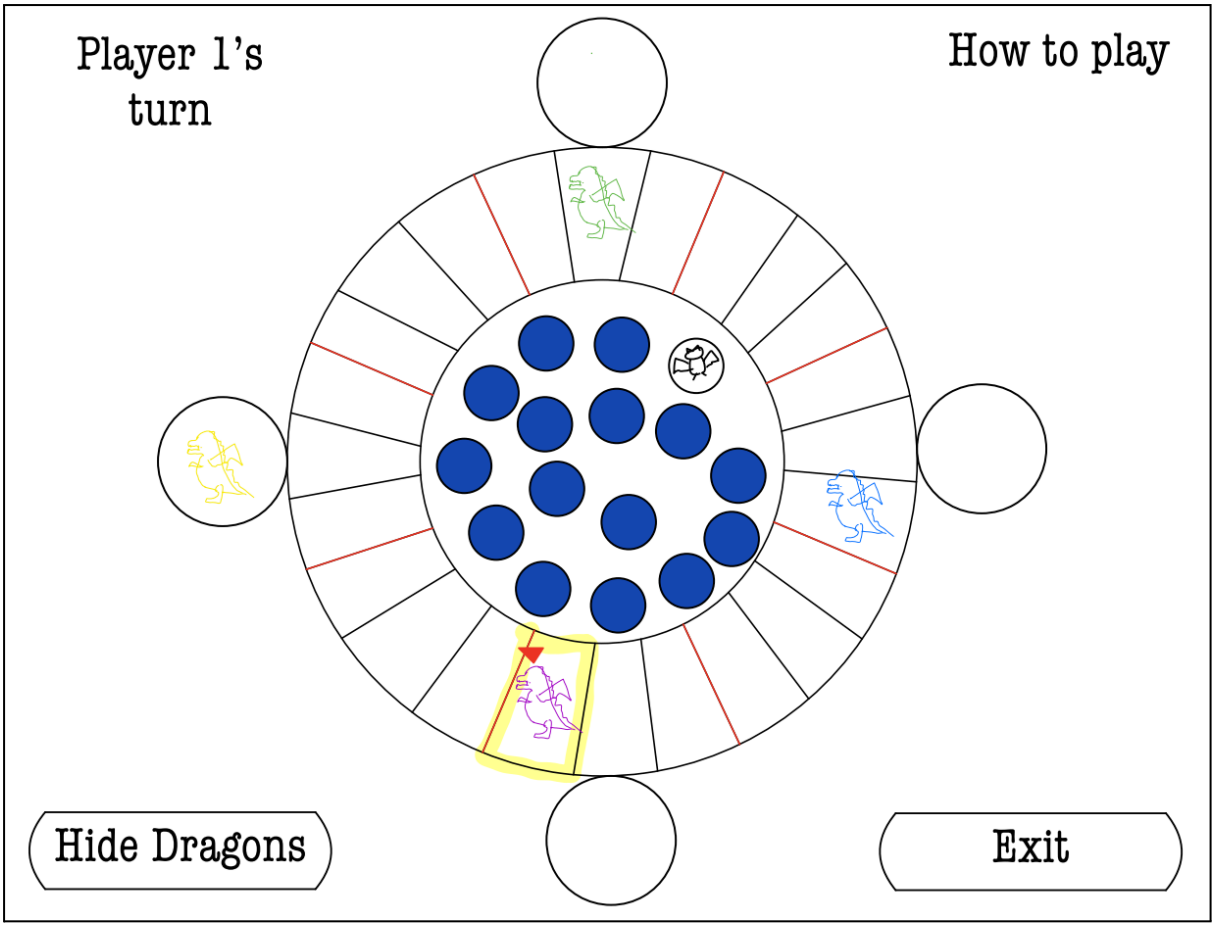
An option to have different board configurations, e.g., with different numbers of Volcano Cards, not all Cards having three tiles.

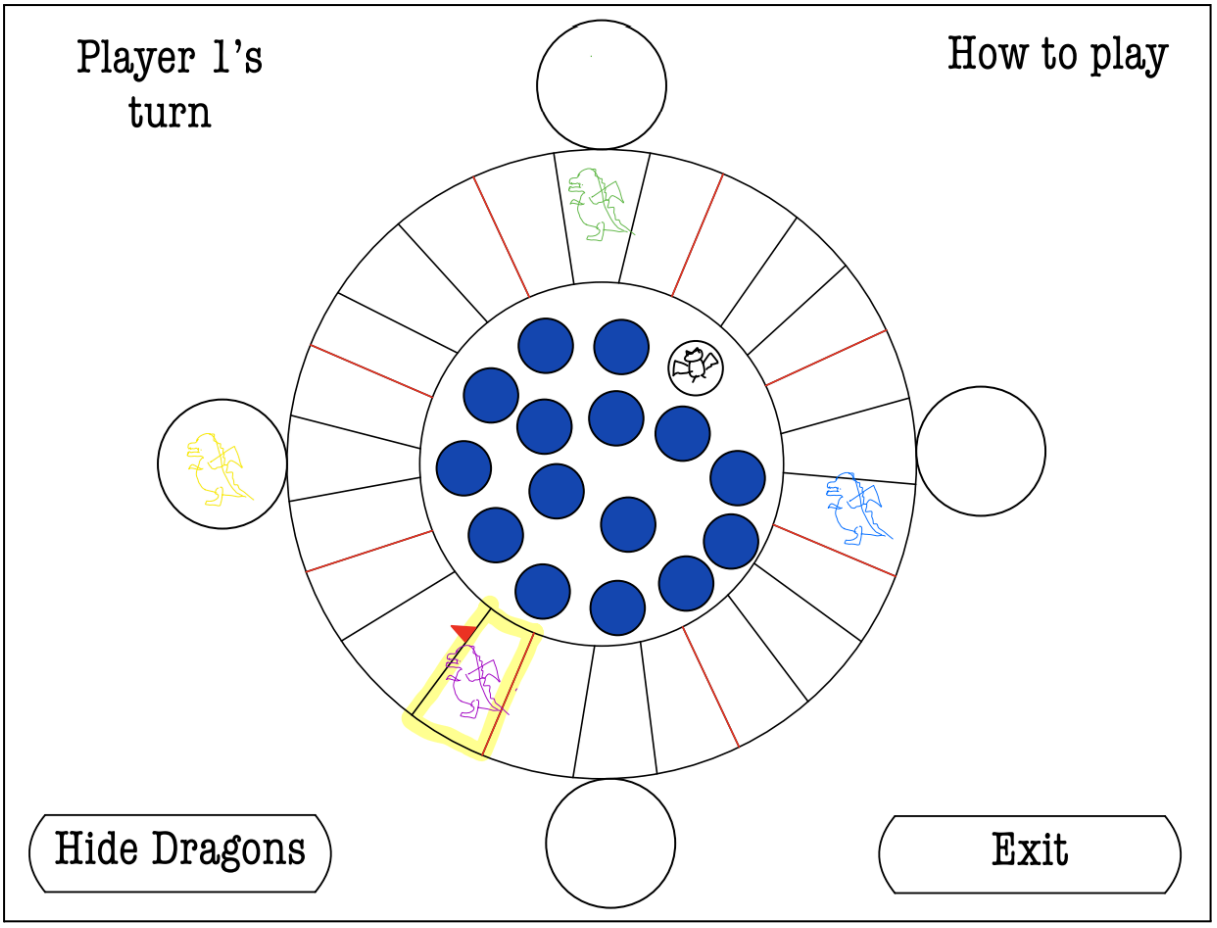
This domain model differs from the base game, as it is possible for the Board to have more or less Volcano Cards per Board. The multiplicity has a minimum of four MapPieces as there are four Cave Cards at a minimum. In addition, since it is now possible to cards with less or more than three tiles, it must also be possible for a singular MapPiece to contain up to all four players.

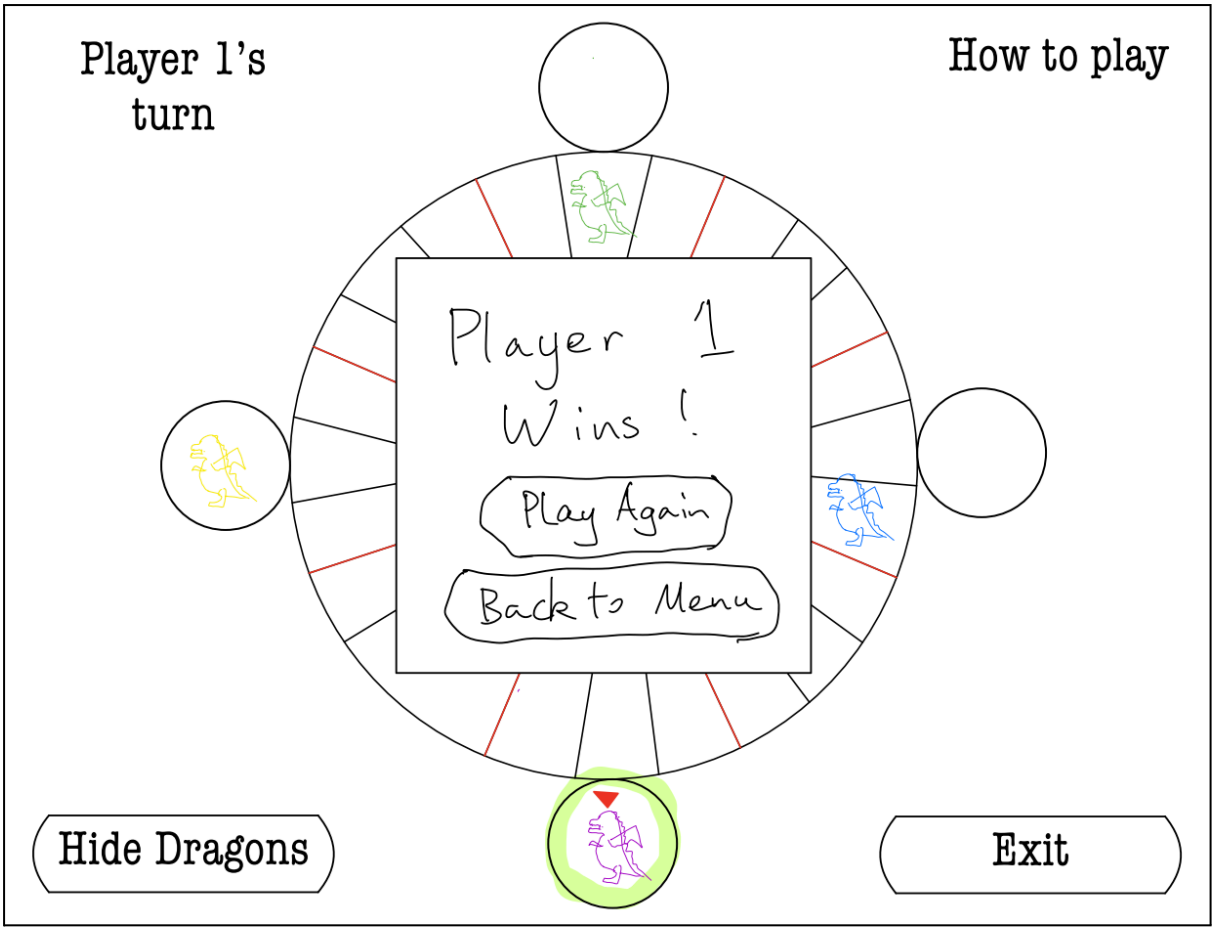
**Basic UI Design**




Player 1's turn
How to play
Hide Dragons
Exit


Player 1's turn
How to play
Hide Dragons
Exit


Player 1's turn
How to play
Hide Dragons
Exit


Player 1's turn
How to play
Player 1 Wins !
Play Again
Back to Menu
Hide Dragons
Exit