FIT3077 Sprint 1

Table of Contents

Team Information Document

Team Name: SANdY (d for Development)

Team Membership

Student Name	Contact Details	Technical and Professional Strengths	Fun Fact
Selsa	Phone: 0403641670 Email: sson0024@student.monash.edu	Python Java Javascript HTML/CSS Communication Collaboration	I have lived in 3 countries
Annie	Phone: 0406034601 Email: ahoo0034@student.monash.edu	Python Java Javascript HTML/CSS Adaptable collaboration	I can crochet and bake
Navya	Phone: 0452625236 Email: nbal0016@student.monash.edu	Python Java JavaScript HTML/CSS Active Listening Problem Solving	I enjoy crafting and building furniture
Yokabit	Phone: 0469290048 Email: yfes0001@student.monash.edu	Python Java CSS Communication Creativity	I like to swim

Team Schedule

The team's regular meeting schedule consists of meetings on Monday at 3:30pm and Friday at 9:30am every week. During these meetings, we aim to discuss project progress, address any challenges and plan upcoming tasks for the specific sprint. In terms of workload distribution and management within the team, we have implemented a system of assigning story points to each task and allowing team members to self-assign tasks in a fair manner based on its story point. This approach ensures that everyone has a clear understanding of their responsibilities and allows for flexibility in task allocation based on individual expertise.

Technology Stack and Justification

Programming language: Java

Our team has experience with Java, making it a natural choice for game development. Java is widely used and provides features for building scalable applications. We aim to utilise the Java Development Kit (JDK) due to the necessary tools and libraries it provides. We also did consider using Python as our programming language as we all were familiar with using classes in Python but we thought creating the UI would be difficult with Python as none of us have created a system like this using Python before, hence we decided to proceed with Java as our programming language.

GUI: JavaFX

JavaFX is a recent, but versatile GUI framework that integrates easily with Java. It provides a range of UI controls, allowing us to create visually appealing and interactive user interfaces. We plan to leverage tools such as ControlsFX for additional UI components and ValidatorFX for input validation, enhancing the usability of our Web App. After considering Swing as an alternative option, we decided to implement FX instead as it is a more modern version with better features.

IDE: IntelliJ

IntelliJ IDEA is a great IDE to support Java game development and all members of the team have used it previously. It has a wide range of features, including code refactoring, debugging tools and can be used in conjunction with Git, which will streamline our development process and enhance productivity. Our team did consider using VSCode as our IDE as it is one we all have used in the past and found it quite useful, but we decided with IntelliJ because it was specific for Java meaning it has in-built Java compilers, easing our coding experience.

Version control: Git

Git provides the opportunity for the team to collaborate, while efficiently managing code changes and tracking project history and progress.

Testing: JUnit

JUnit is a testing framework for Java that will allow the team to write and execute unit tests effectively. By incorporating JUnit into our development process, we can ensure the reliability and quality of our codebase through automated testing.

Documentation: Javadoc

Javadoc is a documentation generation tool for Java code that creates comprehensive and well-structured documentation. The team has used Javadoc before and leveraging it will enhance code readability and facilitate future redevelopments.

User Stories (with Acceptance Criteria)

Epic 1: Initialisation and Game Setup:

1. As a player, I want to be informed about the game rules and mechanics before starting the game so that I understand how to play the game.
    - Have a confirmed set of rules for the game
    - Split the game rules into small sections so the it is easy to understand
    - Display the rules before the game starts
2. As the game developer, I want to set up the game board so that there is a game that can be played.
    - Have a generator to form the board
    - The game board is displayed in the game
    - Validate that each player is assigned a designated starting cave
3. As a player, I want to start the game with the dragon in the designated cave, so that I have a clear starting point for the race around the volcano.
    - Verify that the dragon is correctly positioned at the designated starting cave for each player
    - Display the dragon in the cave on the game board
    - Display the colour of dragon cave
    - Display the colour of the dragon
4. As a game developer, I want to be able to have a home page so that players have a Landing Page to go to when they first open the program or after they have finished a game.
    - Have the name of the game displayed
    - Have images of the game in the background
    - Have buttons - “Play”, “Options”
    - home page is the first screen shown when the game is opened
5. As a game developer I want the dragon cards to be randomised at the beginning of the game so that there is an arbitrary setup of cards so that the player will have to remember the card layout during the game and not before the game starts.
    - Use a randomiser to randomise the cards
    - There should be no predetermined sequence or pattern to the arrangement of dragon cards at the beginning of each game
    - The randomization algorithm must be consistent across game sessions to ensure fairness and consistency in gameplay

Epic 2: Gameplay

6. As a player, I want my dragon’s movement to be based on memory and strategy so that I can plan my moves ahead of time.
    - The tokens should be left in the same place for the duration of the game so that players can apply memory and strategy to win the game
7. As a player, I want to be able to uncover dragon cards and move my dragon according to the quantity shown so that I can advance around the volcano.
    - Players should be able to uncover and recover dragon cards when it’s their turn throughout the game

    - The quantity of tokens on the card should determine the number of spaces the player can move their dragon
    - The token cards are the only way to move the dragon around the volcano
8. As a player, I want the game to have dynamic elements, including challenging dragon cards, so that each game is unpredictable.
    - The game should have a variety of dragon cards to make it challenging and unpredictable
    - Dragon cards should be shuffled after each game to make it fair
9. As a player, I want the interface to be visually appealing and clear so that the gameplay is easy to understand and follow.
    - The graphics and design elements should be clear for all players to understand
    - Important information, such as dragon movement options and the uncovering of dragon cards, should be displayed at the beginning of the game
    - The interface should be easy to navigate and minimise confusion for players during gameplay

Epic 3: Characters

10. As a player, I want to be able to choose my dragon so that I can identify where I am on the game board.
    - Players should be able to select any dragon out of the 4 dragons available at the start of the game, given that it is not taken by another player
    - Players should be able to identify their dragon on the board at all times of the game
11. As a player, I want to be able to see where my dragon is positioned on the game board so that I know how close I am to winning.
    - At the start of the game, the players' positions should all be at their respective cave
    - Player position should be determined by the token card that gets selected by the player
    - When winning the game, the winning player’s position should be back in their cave

Epic 4: Winning Scenarios

12. As a player, I want the game to end when a player’s dragon travels around the volcano and reaches their cave again so that I can either stop playing the game or start a new game.
    - When a player's dragon successfully travels around the volcano and reaches their cave again, the game should recognize this as the winning condition
    - Ensure that the game provides clear feedback to the player when the winning condition is achieved, such as displaying a victory animation
13. As a player, I want a winning page to appear so that I know that a player has won the game.
    - The winning page should be displayed automatically when a player has won
    - The winning page should clearly indicate that a player has won the game

14. As a player, I want the winning page to have two clickable buttons so that I am able to click on the button to be redirected to another page.
    - The winning page must contain two distinct clickable buttons
    - Provide visual feedback (e.g., button highlighting or animation) to confirm button selection when clicked
    - Both buttons should be prominently displayed on the winning page, ensuring they are easily visible and accessible to the player
15. As a player, I want the winning page buttons to say “Play Again” and “Home” so that I can select either buttons to go back to the home page or I can play a new game.
    - The button to play again must be labelled as "Play Again"
    - The button to return to the home page must be labelled as "Home"
    - Both buttons ("Play Again" and "Home") should be clearly visible and accessible on the winning page without the need for scrolling
    - Clicking on the "Play Again" button should initiate the start of a new game with the same game settings as the previous one
    - Clicking on the "Home" button should navigate the player back to the homepage of the game
16. As a player, I want the winning page to have the name of the player that won so that I know who has won the game.
    - Verify that the winning page prominently displays the name of the player who has won the game
    - Confirm that the displayed player name is accurate and corresponds to the actual winner of the game
    - Ensure that the player's name is clearly visible and legible, using appropriate font size, colour, and placement on the winning page
    - Validate that the player's name is dynamically updated based on the winner of each game, reflecting the correct player's name each time a game is won
    - Test the winning page with multiple players winning games to ensure that the correct player's name is displayed each time
17. As a player, I want the game board to be blurred so that the winning page is my main focal point.
    - Verify background screen is blurred
    - The blur effect is applied uniformly across the entire game board, making it visually distinct from the winning page
    - Colours of the board is dull
    - Winning page colours are brighter
    - Confirm that the blur effect does not affect the visibility or functionality of the winning page or any clickable buttons on it
    - Validate that the blur effect is implemented smoothly and does not cause any performance issues or glitches in the game interface
    - Ensure that the winning page becomes the main focal point of the player's attention, with the blurred game board serving as a background element

Epic 5: Multiplayer

18. As a player, I want to be able to choose whether I play the game with 2 players or 4 players so that I can play together with my friends.
    - Promot should be provided at the beginning of each game

    - The prompt should ask the players to select the amount of people playing in the game
    - Amount of players selected should not be less than 2 or greater than 4.
    - The game should enable each player to select a dragon.
19. As a player, I want to be able to name my character so that I know which dragon is mine.
    - Prompt players to enter the name they want for their dragon before beginning the game but after selecting the number of players
    - Provide a text box for each player to enter their name
    - The number of times the name entry is displayed on the screen must be equal to the number of players in this round of the game
20. As a player, I want to be able to choose which colour my dragon is so that I can connect and be more engaged with my character.
    - Prompt players to enter which colour they want for their dragon before beginning the game but after selecting the number of players
    - Provide four colour options for each player to choose from
    - The number of times the colour selection is displayed on the screen must be equal to the number of players in this round of the game

Epic 6: Extensions

- Potion - one of the cards will have a potion symbol on it so when it is revealed, the dragon moves to the next position with the same symbol on it
    - Lucky potion - when the player reveals the lucky position card, their dragon moves to the last position on the game map with the animal on the volcano card they are currently standing on.
    - Jinxed potion - when the player reveals the jinxed position card, their dragon moves to the first position on the game map with the animal on the volcano card they are currently standing on.
- Survivor mode - where the volcano builds up each turn and then eventually erupts which shuffles the chit cards

Our group has come up with two potential extensions of the base game rules.
One of which is the Survivor Mode. This is where the lava inside the volcano in the centre of the game board slowly rises each turn until it eventually erupts. When this happens, the token cards in the centre of the game board shuffles. This complicates the game as it is based on memory of where certain token cards are so when the token cards are shuffled, it will take the dragons longer to reach their cave again. There will be two modes for the game; Regular Mode and Survivor Mode. The mode selection will be done during the setup of the game, after the name and colours have been chosen for each dragon.
Our second extension idea is to have two of our token cards represent ‘Potions’. We have proposed to have two potions; a ‘Lucky’ Potion and a ‘Jinxed’ Potion. When the Lucky Potion token card is revealed, their dragon moves to the last position on the game board with the animal on the volcano card they are currently standing on. When the Jinxed Potion card is revealed, the player’s dragon moves to the first position on the game board with the animal on their current volcano card.

21. As a player, I want to be able to pick up token cards after they have been reshuffled so that the game continues just like before the volcano eruption.
    - When the volcano erupts, token cards should be randomly reshuffled while they are faced down
    - When a token card is selected, the card should be flipped to reveal the image on the other side of the card
    - The image must stay revealed until that players turn is complete
    - At the end of a players turn, all the token cards which have revealed, must be flipped over to hide their image again
22. As a player, I want to be able to move on the board according to the results of the potion cards so that the potion either sets me ahead or behind the other players.
    - When a lucky potion card is revealed, move that player's dragon to the last position on the game board with the animal on the volcano card they are currently standing on
    - When a jinxed potion card is revealed, move that player's dragon to the first position on the game board with the animal on the volcano card they are currently standing on

3. Domain Model Design

Domain Model Link

https://drive.google.com/file/d/1pQ46wW2lqtXnYOcWEVM_O4z3jWjNneMC/view?usp=sharing

Justification for each chosen domain entity and their relationships

List entities:

- GameBoard
  - Represents the environment of the game, including locations like Volcanoes and Caves
- Player
  - Represents a participants in the game that interacts with the Gameboard
- TokenCard
  - Represents the type of cards that can be flipped in the middle of the board
- Dragon
  - Players will interact with Dragons so that they can move around the board
- Volcano
  - Part of the extension as it represents a dynamic element on the Gameboard
- Cave
  - Represents the starting and ending points for each dragon
- Main
  - Needs to be run for the game to begin
- GameLogic
  - Defines rules and actions of the game, including movement and card interactions
- Action
  - Includes moving back/forward, into/out of cave, flip card action, volcano eruption action, reshuffle card action
- Position
  - Indicative of the location of the dragon across the board

The GameBoard entity represents the environment or game world where the gameplay unfolds. It encompasses various locations, terrain features, and elements crucial for player interaction and progression. The GameBoard entity is vital given that it is a main part of the game where players are able to interact and overall play the game. The relationship that GameBoard has with other entities are direct association with Cave, GameLogic, Volcano, Position and TokenCard. The association between the volcano entities is established as the Gameboard provides spatial information about the location of volcanoes. Finally, the TokenCards have an association relationship with the Gameboard given that it contains information about the placement and effects of token cards.

The Player entity represents the participants in the game who interact with the game board and is an instance of the game. Each player is an essential component of the game as they actively engage in gameplay, making decisions and taking actions that affect the game's progression. The player entity has a direct association with both the dragon entity and

GameLogic entity. This was chosen because Players' actions and decisions are governed by the rules and logic defined by the GameLogic entity. GameLogic dictates how players can interact with the game environment, including movement, interactions with other entities, and resolution of game events.

The TokenCard domain entity represents a superclass of the different card options. The TokenCard entity was chosen since it represents the type of cards that can be flipped in the middle of the board. The sub-classes of the TokenCard will inherit the attributes and methods from the superclass. The relationship between the classes is inheritance, this was chosen because it promotes code reuse, as common features can be defined in the superclass and reused in multiple subclasses without duplication as well as minimise code inconsistencies. Furthermore, having sub-classes enables the team to add new methods into the class that are specific for the certain cards.

The dragon domain entity represents a superclass of the player's characters. The dragon entity was chosen since it represents the in-game creature, which inturn will display where the players are positioned on the board. Further it was crucial that a dragon entity was created as it defines how the players move and react based on the cards that will be chosen by the player. Given that the dragon entity is the superclass, the sub-classes will inherit the attributes and methods from that class thus the relationship between the dragon entity and the PinkDragon, PurpleDragon, BlueDragon, GreenDragon will be an inheritance which the four colour dragons shall inherit from the main dragon class. The team has decided to separate the different coloured dragons into different sub-classes since each player will have different positions on the map and the character will be played by different players. Furthermore, the use of inheritance prevents repetitive code from being created which then will reduce the form of errors. Given that codes would not be copied over for each different character this will encourage consistency in the colour dragon code as well as prevent lines of code from being lost or deleted. Moreover, the dragon entity has a direct association with Positions. The relationship between Dragon and Positions is formed to determine the current location of the dragon on the board.

The Cave entities are integral components of the game environment represented by the Gameboard. They serve as starting and ending points for each dragon's journey within the game. The Gameboard contains information about the layout and placement of caves. The direct association relationship was formed between the cave and GameBoard since the cave was established as integral parts of the game environment, alongside other features like volcanoes.

The Volcano entity has been included because our extension involves the eruption of the volcano. This entity represents the dynamic element, the volcano, in the centre of the game board. This explains the direct association relationship with the GameBoard entity as the Volcano will be used as a part of the GameBoard. It is a one-to-one relationship as each game board will have one and only one volcano on it. The association relationship with the GameLogic entity is because of the Volcano eruption being part of the actual game logic hence GameLogic will have a Volcano instance as its attribute.

The Main entity in Java is where the main() method is implemented. This is the class and method that is run first so it will contain the logic which starts an instance of the game. There

is a one-to-one association relationship between the Main entity and the GameLogic entity because GameLogic will contain the majority of the implementation of the game, the Main entity will just initiate the game.

The GameLogic entity represents the logic of the game. It will include all the implementation from the beginning of the game (selecting how many players are playing and selecting characters) to the very end of the game (when a player wins the game). We decided to create this class so that all the main functionalities of the game will be in one class and all other classes interacting with this class will have their own implementation which is part of the logic of the game. The movement of the dragon for each turn as well as the graphics for the volcano eruption will be implemented in other specific entities but an instance of those classes will be created in the GameLogic entity as it is part of the main functionality of the game. GameLogic has an association relationship with Player, Main, Volcano, GameBoard and Action as these are all vital parts of the game logic itself. There is a 2 to 4 relationship on the Players side of the relationship as for each game, there can be a minimum of 2 and maximum of 4 players. There is a one to many relationship on the Actions side because there will always be at least one action in the game like the CaveAction and FlipCardAction.

The Action entity represents all the events which take place in the game. Action will be an abstract class with all the implementation for the actions in each subclass which are ReshuffleAction, FlipAction, CaveAction, VolcanoAction and MovementAction. The MovementAction class is for moving forward or backwards to other volcano cards. The CaveAction represents the action of moving into or out of a cave. The VolcanoAction represents the volcano eruption extension and the ReshuffleAction will be used after the VolcanoAction to reshuffle the token cards in the centre of the game board. Although ReshuffleAction and VolcanoAction will be performed one after another, our team has decided to separate them because future extensions may also use the ReshuffleAction without needing the eruption from VolcanoAction. Our team decided to make Action an abstract class because all the action classes must have the same main method implemented which will be the execute() method. Since all the execute methods in the action classes have the same name, our implementation in the GameLogic entity will be simplified as we can use polymorphism to perform the execute() method. Furthermore, the sub-classes give us the ability to add other methods into the classes that are specific to that action. The Action entity has an association relationship with GameLogic because these actions are a pivotal part of the game logic and they are the main events that will be taking place in the game.

The Position entity represents the spatial information about the locations of the dragons on the game board. Each position on the game board will be a separate instance of the Position entity. This is why there is an association relationship between GameBoard and Position. The specified multiplicities are because each position can only be part of one game board and the gameboard will have many positions but it must have at least one position. The dragons will be located on these positions hence the association relationship between Dragon and Position. Each position can have zero or multiple dragons and each dragon can only be in one position hence the specified multiplicities.

Domain Model Design Choices

Throughout the process of creating the domain model, various designs were explored to ensure that the concept of object oriented principles were applied extensively. Our current domain model utilises inheritance to enable code reusability, providing the possibility of extensions in future development. In the earlier design stages, it was agreed upon to have one entity for Actions, however after further discussion and analysis of the requirements, we concluded that having one entity that contained all the actions was not feasible as it could create issues when utilising them in various classes. Hence why in the current model, there is a basic Action class which all other actions extend from. The main idea behind the specific choices in the domain model was just to make extensions easy to apply throughout development. In the domain model, two extension requirements were implemented which slightly impacted the design choice of the TokenCard entity and Volcano entity. As part of the extension requirements, two additional types of tokens were added, Jinx Potion token and Lucky Potion token which give players a debuff or buff respectively, hence the additional token entities that extend from TokenCard. As mentioned before, the different types of tokens extended from the TokenCard class to enable code reusability and efficient implementation. Another extension requirement that has been implemented is that after a certain amount of turns, the volcano will “explode” and reshuffle the token cards. To enable the addition of this extension, we have added a ReshuffleAction entity and VolcanoAction entity to represent the two main components of this extension. We also connected the Volcano entity to the GameLogic entity to give it functionality in terms of the gameplay.

Assumptions

Some assumptions made while designing the domain model mainly revolved around the game board itself. Based on our understanding of the brief and game description, we assumed that there can never be less than 2 people playing the game at a time and no more than 4 of course due to the limitation of the gameboard. It is because of this assumption that the cardinality of the association that connects Player and GameLogic has a 2..4 to 1 multiplicity and the association that connects Player and Dragon has a 2..* to 2..* multiplicity. Likewise, the cardinality between Cave and GameBoard has 2..4 to 1 multiplicity as the number of caves available on the game board is directly correlated to the number of players, therefore the assumption impacts the design of the Cave and GameBoard entity.

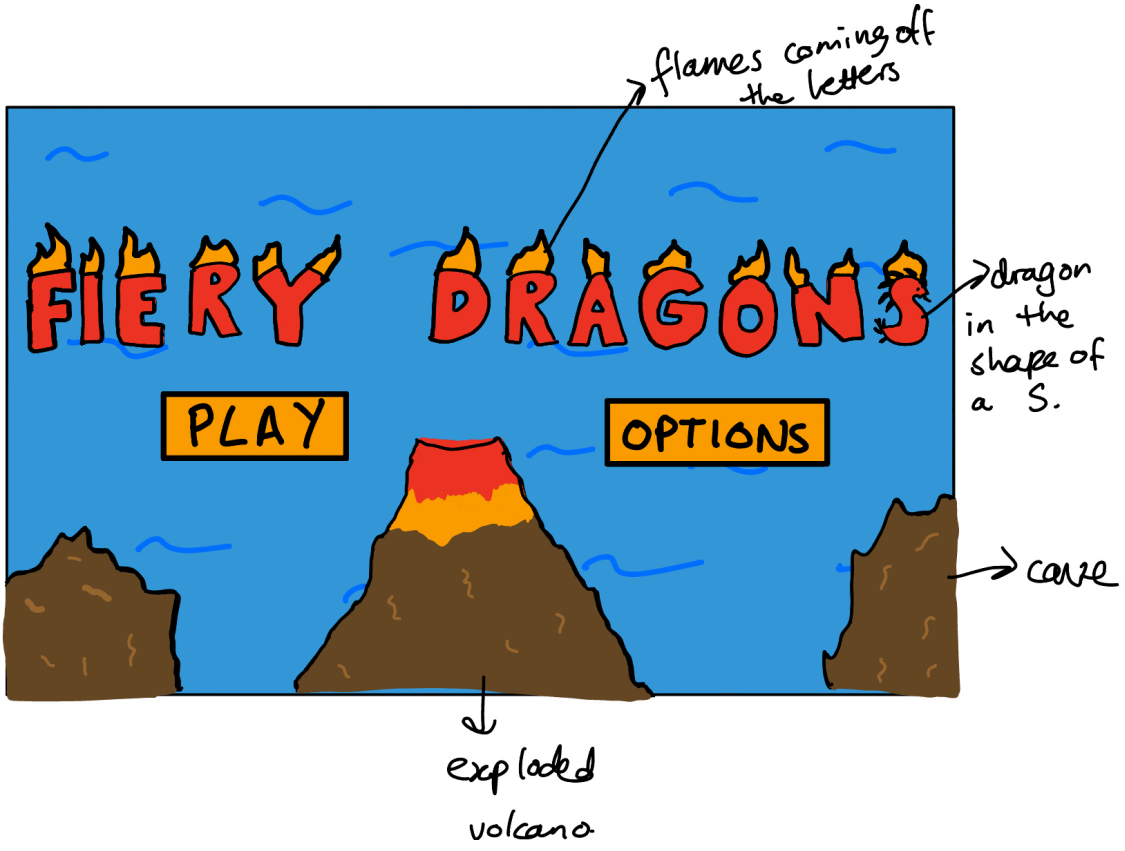
Justification for Modelling

Inheritance between TokenCard and other cards, such as DragonCard, establishes that it will inherit properties and behaviours from TokenCard, while also having its own unique characteristics. Hence, each card class has a single responsibility, making the code more maintainable and easier to understand. Additionally, in terms of the Open-Closed principle, this design allows for new card types to be added without modifying the existing classes, reducing the risk of bugs and facilitating code extension. This also applies to the Action class which implements inheritance towards several actions such as CaveAction. In terms of the GameBoard, through using association, it has been split into Cave, Volcano and TokenCard entities. This reflects a clear separation of concerns as the entities are responsible for different aspects of the GameBoard, helping to maintain and extend code when needed.

Through using association, the domain model can be easily extended to include new entities and relationships without modifying existing ones.

4. Basic UI Design Draw

Prototype 1: Set up (Homepage)



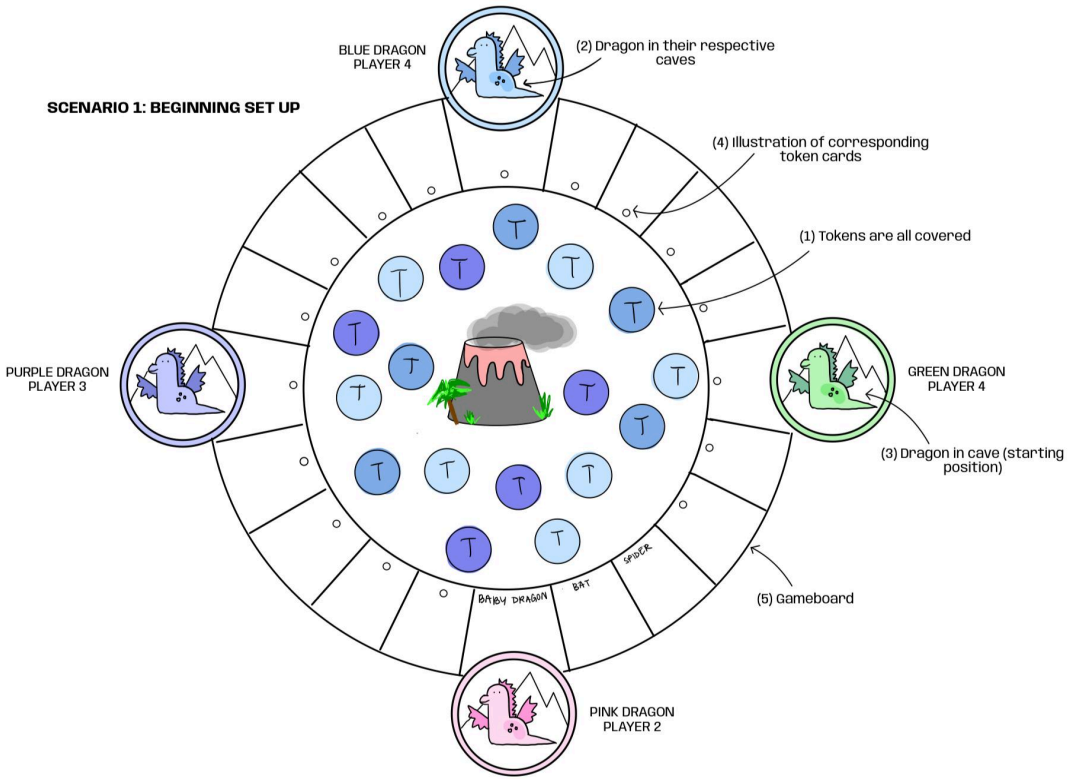
Prototype 2: Selecting Players

Prototype 3: Name of Players

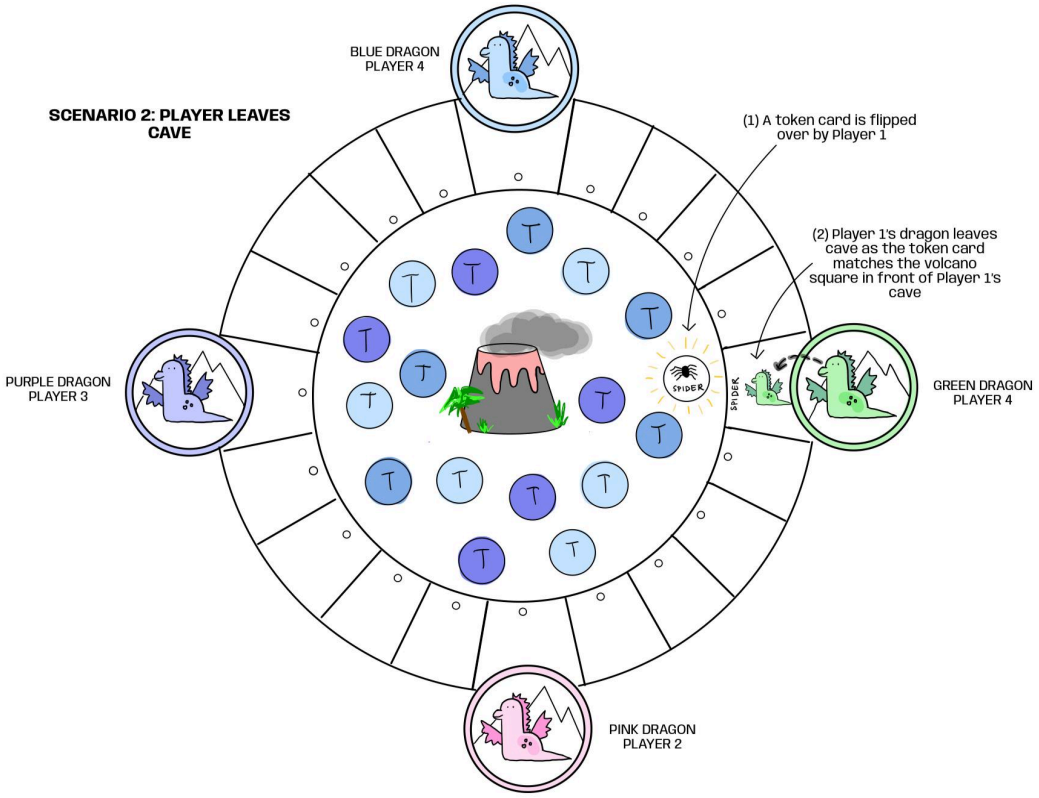
Prototype 4: Character selection for each player

Prototype 5: Selecting the Mode

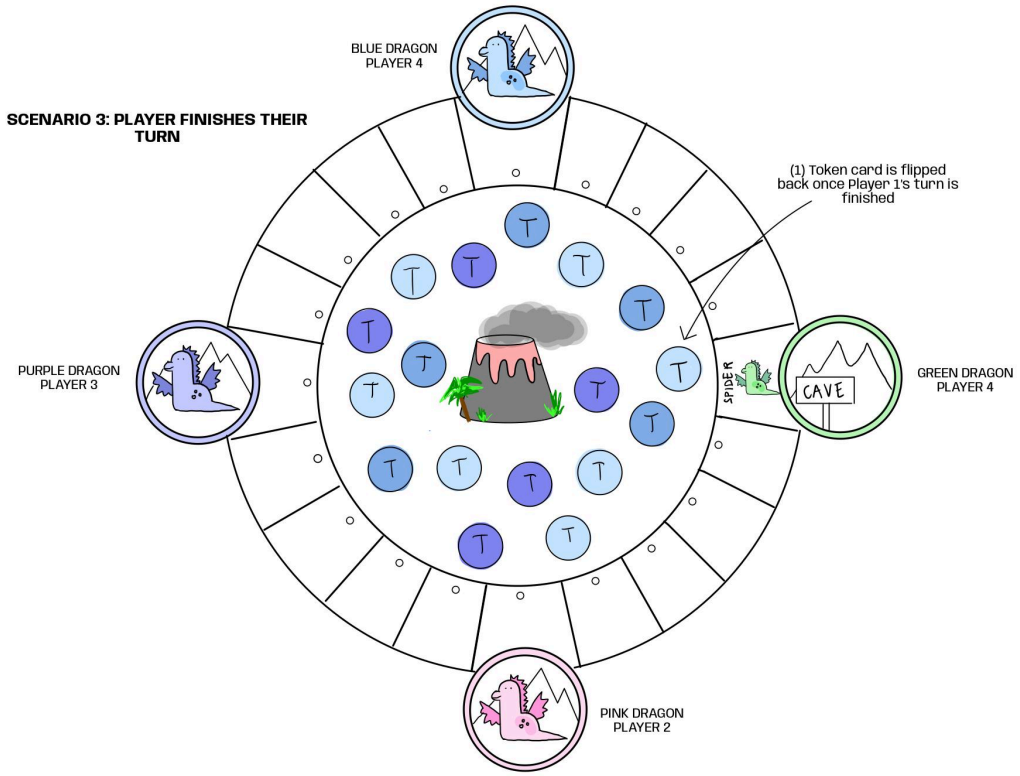
Prototype 6: Dragons start inside Caves



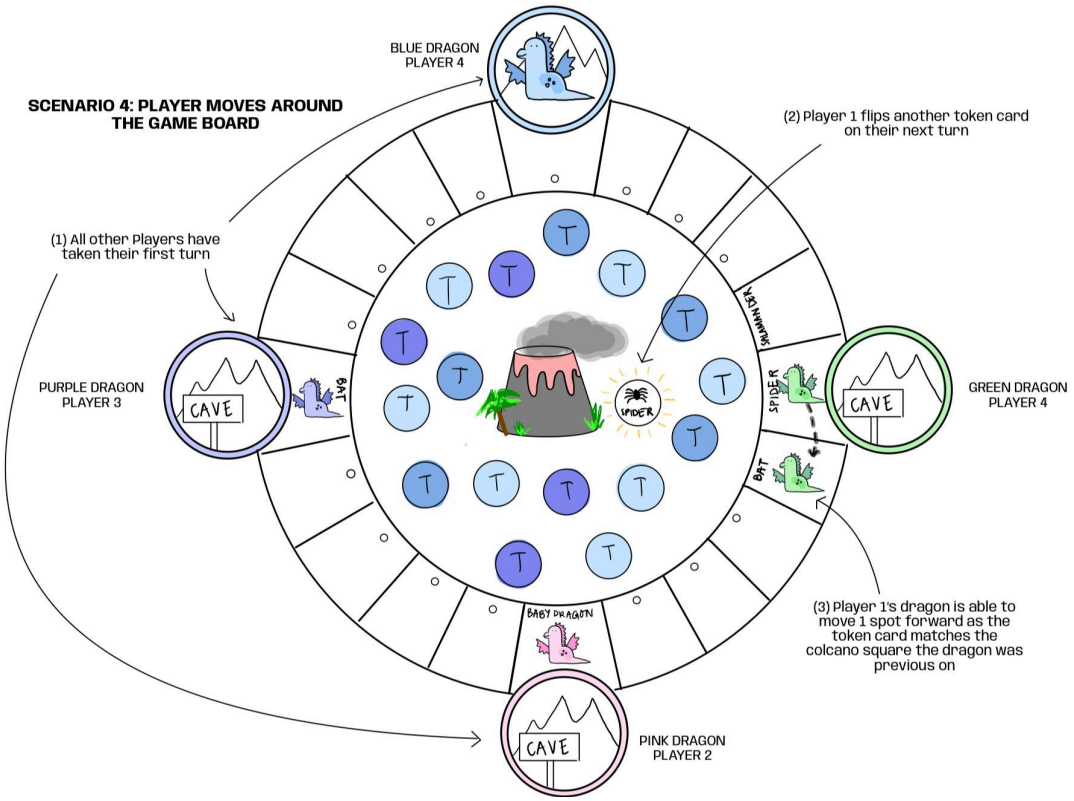
Prototype 7: Player leaves Cave



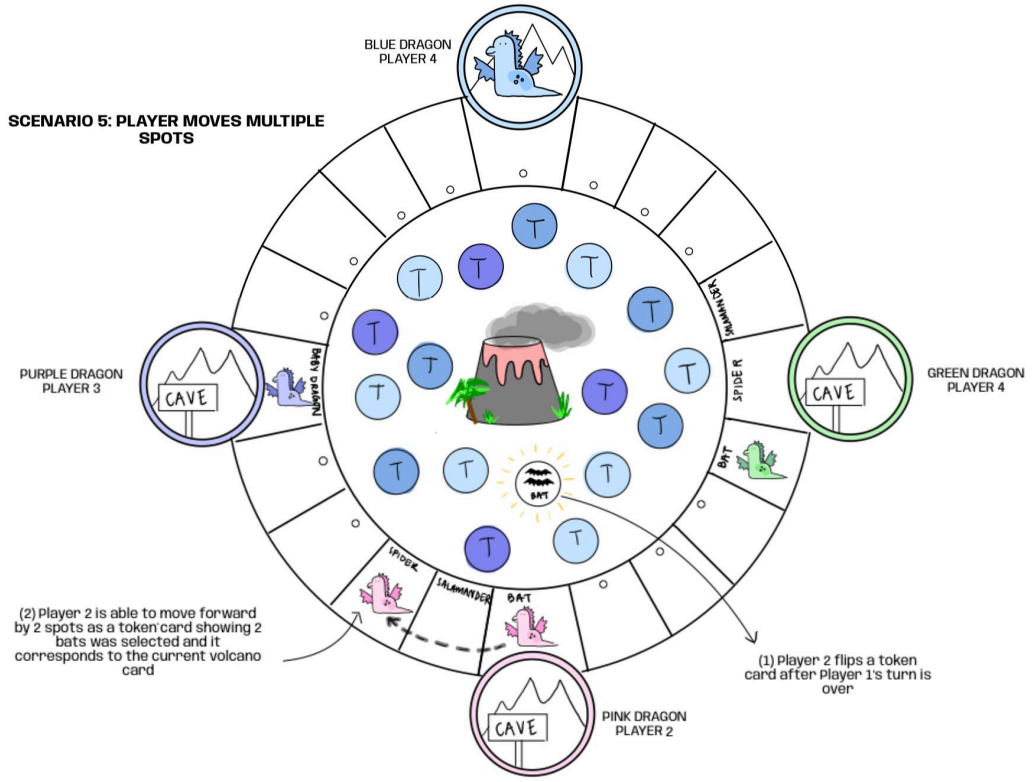
Prototype 8: Player finishes their turn



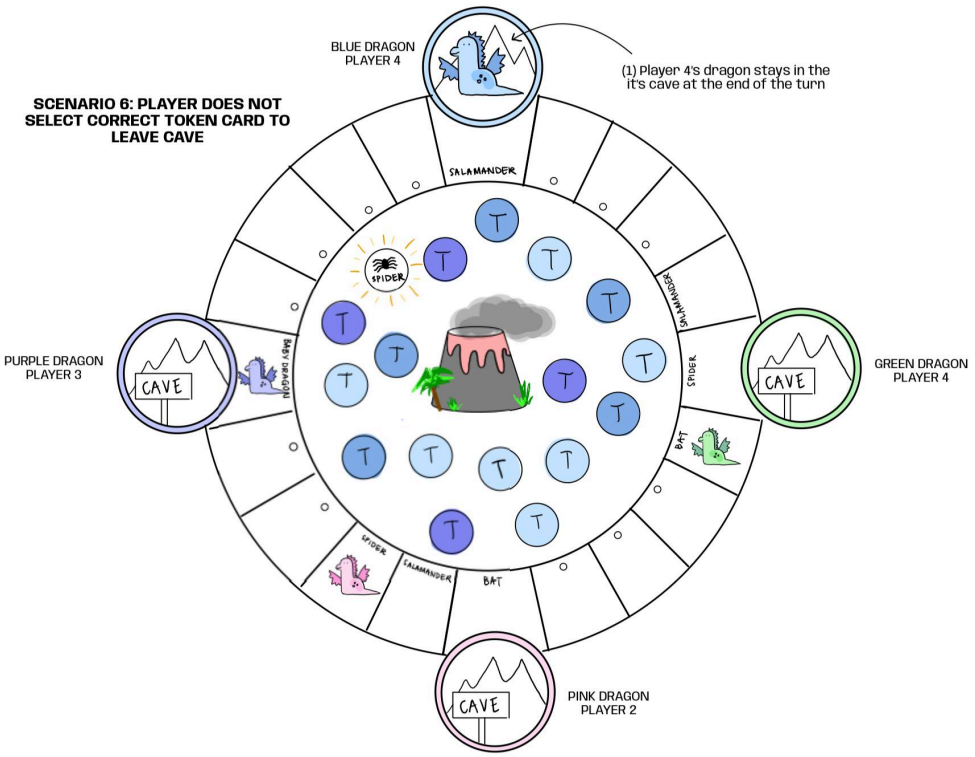
Prototype 9: Player moves around Gameboard



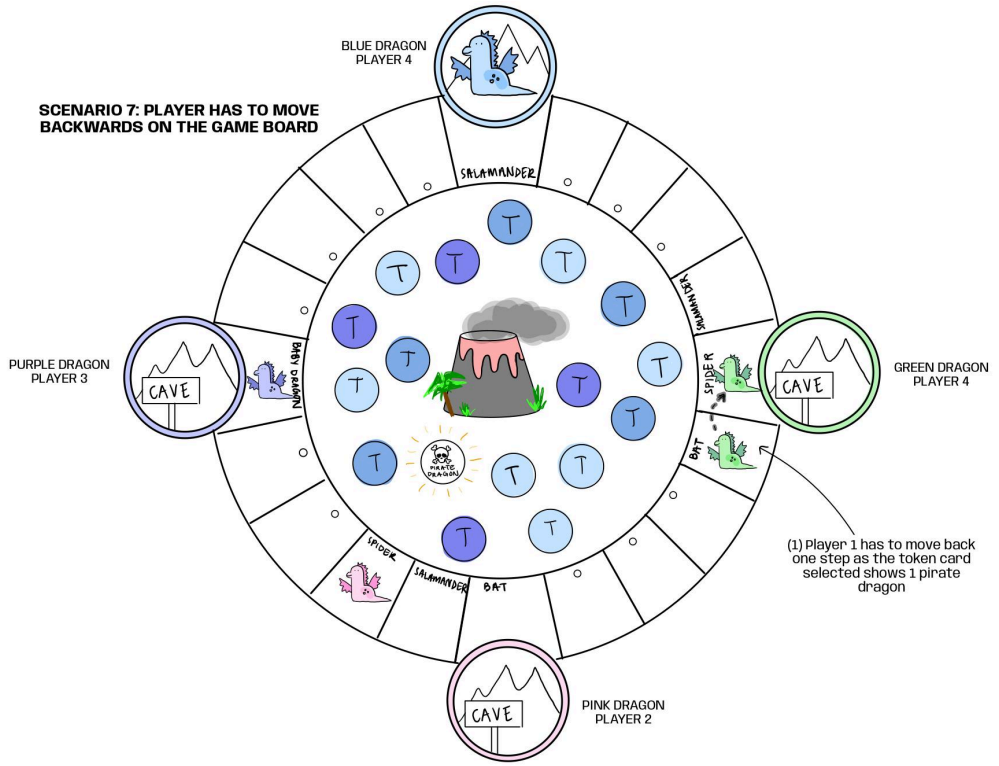
Prototype 10: Player moves multiple steps



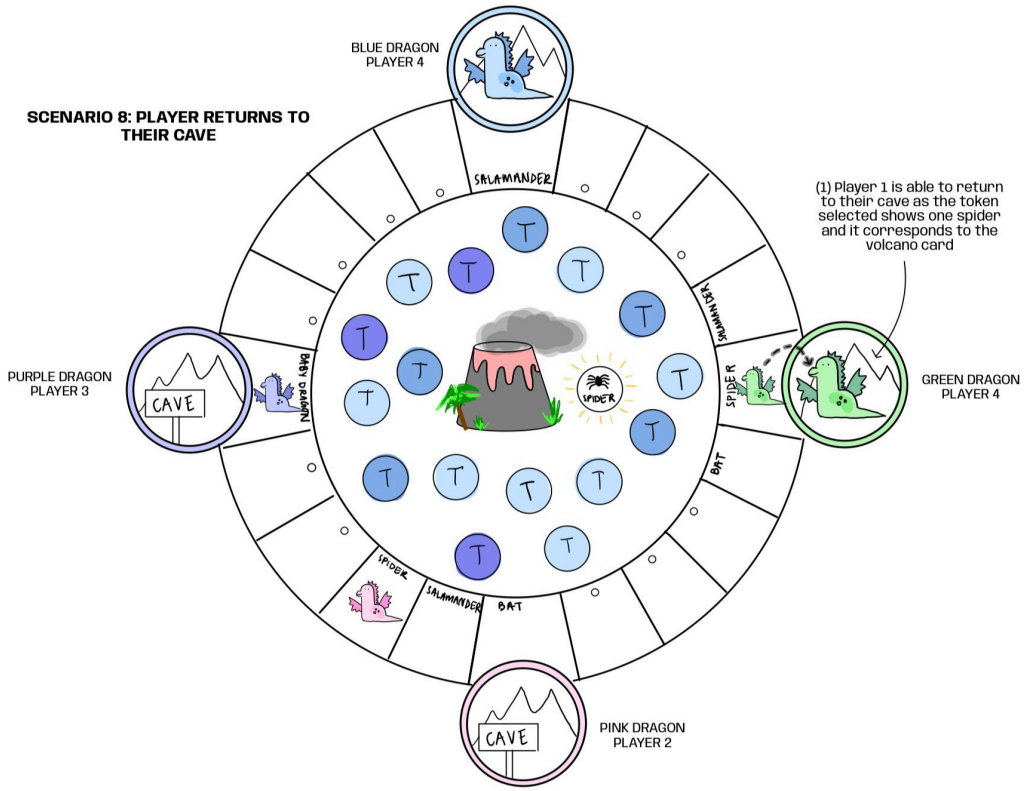
Prototype 11: Player does not select correct token card to leave cave



Prototype 12: Player moves backwards on the game board



Prototype 13: Player returns to Cave



Prototype 14: Winning Situation