LUMINARY

FIT3077 Software Engineering: Architecture and Design

Fiery Dragons Game Software

Sprint One

Luminary Team
[MA_Tuesday12pm_Team002]

Written by:
Koe Rui En
Tay Ming Hui
Wong Jia Xuan

Contents

1. Team Information

1.1 Team Name and Team Photo

The words ‘Luminary’ refer to a person who holds significant influence as a leader in a specific field. In the context of a game development team, “Luminary” would imply that a team aspires to be a shining example and also describe a team that strives for excellence and innovation. Hence, Luminary is the official name of our team.

Diagram 1.1.1: Tay Ming Hui, Wong Jia Xuan, Koe Rui En (left to right)

1.2 Team Membership

Introducing our team, Luminary, the masterminds behind the Fiery Dragons board game project, dedicated to crafting an unforgettable gaming experience for players all around the world.

1.2.1 Meet The Team

The basic information of each team member, along with their technical and professional strengths are documented below. Additionally, intriguing fun facts about each team member will also be revealed.

Name	**Email Address**	**Contact Number**
Koe Rui En	rkoe0003@student.monash.edu	+6010-2978665
Tay Ming Hui	mtay0031@student.monash.edu	+6019-8186824
Wong Jia Xuan	jwon0164@student.monash.edu	+6017-7531317

Table 1.2.1.1 Team Contacts

Name	**Technical and Professional Strengths**	**Fun fact**
Koe Rui En	Technical strengths: ● Proficient in the Java and Python languages ● Skills in version control systems such as GitLab for the team's project. ● Knowledge of agile methodologies	Passion for playing adventurous-themed games
	Professional strengths: ● Proficiency in coordinating tasks and managing timeline ● Commitment to precision in tasks	
Tay Ming Hui	Technical strengths: ● Proficient in the Java and Python languages ● Skills in version control systems such as GitHub and GitLab for the team's project ● Experienced in Front-end project	Excited to learn GUI
	Professional strengths: ● Ability to complete tasks fast and with quality ● Fast problem solving skill	

Wong Jia Xuan	Technical strengths: ● Proficient in the Java and Python languages ● Skills in version control systems such as GitHub for the team's project.	I don't like the human species.
	Professional strengths: ● Integration of software and hardware	

Table 1.2.1.2 Team Technical and Professional Strengths and Fun Fact of Team Members

1.3 Team Schedule

This section will document our team's meeting and work schedules, along with workload distributions and management within our team.

1.3.1 Meeting and Work Schedules

Our team will develop a series of regular meeting and work schedules to ensure that we remain on track and meet our assigned timelines. Therefore, we will plan the most suitable meeting and work times to ensure all team members can make progress efficiently without burning out, thereby producing high-quality software work.

The table below displays our team's regular meetings and work schedules. This table would assist us in planning and managing our work.

Schedules	**Day**	**Time**	**Venue / Platform**
Meeting	Friday	10:00 am - 11:00 am	Online Zoom
	Tuesday	12:00 pm - 12:15 pm	Tutorial Room 6302
Work	Tuesday	12:15 pm - 2:00 pm	Tutorial Room 6302
	Monday, Wednesday to Saturday	3:00 pm - 5:00 pm	Room 9306

Table 1.3.1: Team's Regular Meeting and Work Schedules

1.3.2 Task Allocations

Our team will distribute workload accordingly by conducting regular meetings to determine tasks that need to be completed in each sprint. Hence, we will allocate tasks based on voluntary assignment of tasks according to Scrum practice. Team members are also allowed to express their preference for specific user stories or features before each sprint starts. Additionally, team members are encouraged to volunteer themselves based on their interests and skills.

To manage our team's workload, we will use Google spreadsheets to record the distributed tasks, set planning dates and track progress. For instance, user stories or features to be implemented will be broken down into smaller tasks, which will then to be assigned to specific members, and organised into lists based on their status (to-do, in progress, done and incomplete). This will allow team members to view their assigned tasks, helping them prioritise and manage their tasks efficiently. Furthermore, we will review the team's progress and discuss any challenges faced during implementation in our regular meetings. Each team member can provide a brief update on their tasks during the meetings or via our group communication platform. This will enable our team to identify any issues faced and make necessary adjustments to the workload.

Our team genuinely trusts each other to allocate tasks that best suit our abilities and interests in line with the spirit of Agile values and principles. As stated in the "Individuals and Interactions" core value of the Agile manifesto, our team will communicate frequently to understand each other's tasks and foster collaboration. When necessary, team members will assist each other when difficulties arise during task implementation.

1.4 Technology Stack and Justification

1.4.1 Comparison of Choices of Programming Languages, APIs and Technologies

In this section, we will explore the programming languages, APIs, and technologies that our team plans to utilise after comparing their benefits and drawbacks. We will provide justifications for our choices and examine how they align with our team's current expertise. Additionally, we will also identify any challenges in expertise that may require support from the teaching team.

A) Programming Languages, APIs

Programming language and GUI	**Advantages**	**Disadvantages**
Java with Java Swing	- **Mature and Stable** Swing has been part of the Java platform since its early days, ensuring stability and reliability, making it a dependable choice for building our project. - Cross-Platform Compatibility Swing can run on any platform that supports Java. This allows us to run our project for different operating systems. - **Rich Component Set** Swing offers a comprehensive set of GUI components including buttons, text fields, tables, etc. These components are highly customizable and can be easily integrated into complex UI layouts. - **Integration** Swing integrates seamlessly with other Java technologies and libraries.	- **Complexity** Building complex layouts in Swing may require a significant amount of code. While Swing's API design is powerful, it can lead to verbose and intricate code, making development more time-consuming and error-prone. - **Performance** Swing applications may not perform well especially for graphics-intensive or resource-intensive tasks with limited hardware resources.
Java with JavaFX	- **Modern UI** JavaFX offers smooth animations, rich visual effects, and support for modern design principles. - **Allowing FXML** JavaFX allows to define UI layouts	- **Steeper learning curve** JavaFX may have a steeper learning curve compared to Swing. Learning concepts such as FXML and JavaFX's

	using FXML, which separates from UI design from application logic This separation promotes cleaner code architecture, making it easier to maintain and modify UI design. - **Powerful animation capabilities** JavaFX offers powerful animation capabilities for creating interactive and dynamic user interfaces. - **Supports CSS styling** JavaFX supports CSS for styling UI components, providing us a familiar and flexible approach to designing UIs, making it easier to achieve a consistent and polished look across an application.	event-driven architecture can take time and effort. - **Less widespread adoption** JavaFX has less widespread adoption compared to Swing which results in limiting the availability of third-party libraries, tools, and resources.
Python with Python GUI (Tkinter or PyQt)	- **Popular and versatile** Python is a popular and versatile programming language known for its simplicity, readability, and ease of use. Its syntax is straightforward for use. - **Simplicity** Python GUI libraries like Tkinter and PyQt are designed to be straightforward and easy to use, allowing developers to quickly create functional user interfaces. - **Large and extensive third-party libraries** Python has a vast ecosystem of third-party libraries and frameworks, including GUI toolkits which provides additional features. - **Rapid development and prototyping** Python's simplicity enables rapid development and prototyping of GUI applications. With Python, we can quickly iterate on our ideas and test them without getting bogged down by boilerplate code or complex syntax.	- **Less animation support** Tkinter may not offer the same level of advanced animation features as some other GUI frameworks, such as JavaFX. Achieving complex animations may require more effort and expertise. - **Performance issues for resource-intensive applications** Python may not be as performant as Java for CPU-intensive or memory-intensive applications, particularly in those requiring extensive computations or graphics rendering.

Table 1.4.1.1: Comparison between advantages and disadvantages of Programming Languages and APIs

B) Technologies

i) Version Control

Gitlab is mandatory for this unit. Gitlab is a convenient tool to manage the modification of code and keep track of the changes that have been made during the development. Furthermore, it can record each member's commits and produce a graph for monitoring each member's contribution.

ii) IDE (Integrated Development Environment)

IDE	Advantages	Disadvantages
IntelliJ	- IntelliJ supports more programming languages and frameworks. - Providing refactoring and autocomplete features. - The user interface is more user-friendly.	- Less plugin options but the quality is high. - Some plugins are not free.
Eclipse	- Provides more plugin options. - Free and open source.	- Not providing the refactoring and autocomplete features.

Table 1.4.1.2: Comparison between advantages and disadvantages of IDE

1.4.2 Final Decision

After a thorough evaluation of various language and GUI toolkit options, our team has conclusively decided to proceed with Java with the Swing GUI toolkit for our project.

Our team has experience with Java and Python. Python and its various GUI libraries, such as Tkinter, PyQt, or wxPython, were not chosen primarily because our team's expertise lies more heavily in Java. While Python is a powerful and versatile language, our extensive experience with Java's object-oriented programming principles and game coding makes it the more natural choice for this project. Leveraging our existing Java knowledge allows us to build upon our strengths and potentially overcome challenges more efficiently. While Python GUI libraries are capable, they may not offer the same level of dedicated support and documentation for game development as Java's Swing or other Java-based game development frameworks and libraries.

As for JavaFX, although it is a more modern GUI toolkit for Java with rich multimedia capabilities, our team has limited experience and knowledge with it. Swing's maturity, extensive documentation, and our team's existing knowledge of it reduce the learning curve and potential risks associated with adopting a new technology like JavaFX for this project. However, we acknowledge that JavaFX could be a viable option for future projects or potential enhancements to our current project, as it offers improved performance and a more modern look and feel compared to Swing.

While we are relatively new to GUI development with Java, Swing is a part of the core Java libraries, making it a natural choice for our team to leverage our existing Java knowledge. Swing offers a rich set of components implemented entirely in Java. Its comprehensive set of customizable components and consistent appearance across diverse platforms aligns well with our needs [1], enabling us to create engaging and intuitive user interfaces without the need for extensive custom development. Besides, Swing has been a part of the Java ecosystem for a long time and has amassed complete documentation, tutorials, and community support. This extensive documentation and resources will be invaluable for our team as we navigate through the learning curve of GUI development with Swing. While JavaFX offers improved performance for certain use cases, Swing is generally considered to have adequate performance for most game user interfaces. Since our project's primary focus is on game logic and mechanics rather than heavily graphics-intensive user interfaces, Swing's performance should be sufficient for our needs.

Our team already possesses substantial experience in Java game coding, which will be invaluable for this project. However, we anticipate needing guidance and support from the teaching team specifically for Java Swing development. As we have recently started learning Swing, we may require assistance with effective utilisation of Swing components for game user interfaces and performance optimization strategies. Depending on the project's complexity and specific requirements, we may also seek support from the teaching team on advanced game development techniques, such as game engine architecture, graphics rendering and multiplayer implementations.

To support our development process, we will be utilising IntelliJ as our integrated development environment (IDE) and GitLab as our designated version control tool.

IntelliJ provides excellent support for Java and Swing development, enhancing our productivity and streamlining the development workflow. Meanwhile, GitLab will facilitate efficient collaboration among team members, efficient code management, and tracking of changes throughout the project.

By capitalising on our team's existing strengths and the capabilities of Java, Swing, IntelliJ, and GitLab, we are confident in our ability to create high-quality software that meets our project requirements and delivers an engaging user experience. Furthermore, by proactively seeking guidance from the teaching team in areas where we anticipate knowledge gaps, we aim to overcome challenges effectively and enhance the overall quality of our project.

2. User Stories

A user story is a short narrative that represents the features that the client wants the system to have. It usually follows a simple format:

As a <role>, I want to <desirable action/reaction/state> so that <benefit>

Besides following the format above, crafting a good user story should also apply all aspects of the INVEST criteria, which are Independent, Negotiable, Valuable, Estimable, Small and Testable, to avoid any failure in the implementation of features in future sprints.

Hence, we will strictly follow the format and INVEST criteria user stories The user stories that covered basic Fiery Dragon game functionality as well as team-defined extensions are shown in tables 2.1 and 2.2 below.

A) Basic Fiery Dragon Game Functionality

User Story ID	User Story
US01	As a software developer, I want to produce 4 players, 4 caves, 16 chits, 8 cards to form a circle to represent the physical layout so that players are engaged and immersed in the gameplay experience.
US02	As a software developer, I want to shuffle the dragon cards and spread them out face down in the inner area of the volcano so that this is fair game.
US03	As a software developer, I want to ensure that this game allows 2 to 4 players and the age of the player is between 5 and 99 so that the game can be played.
US04	As a software developer, I want to ensure the game rules are implemented correctly so that the game can be played smoothly without misleading the players.
US05	As a software developer, I want to ensure the interactive components can only interact with the players if it is the player's turn so that the other players won't affect the current player.
US06	As a software developer, I want to cover up all the dragon cards once the player’s turn is over so that other players won’t feel confused.
US07	As a software developer, I want to ensure the dragon(the player’s token) only moves backward on the volcano cards once the dragon has stepped out of the cave so that the player can only go back to his/her cave in a clockwise direction.
US08	As a software developer, I want to ensure that the player only can reach his/her cave with the exact number of moves so that the players can reach their cave according to the rules.

US09	As a software developer, I want to ensure that the game ends once a player reaches his/her cave so that the winner can be determined.
US10	As a software developer, I want to end the player’s turn if his/her dragon lands on an occupied square so that the game can continue.
US11	As a graphic designer, I want the game board to have clear markings for each square, cave, and dragon card so that the movement and positioning are easily understood.
US12	As a graphic designer, I want the size of game components like game board, dragons, volcanos and dragon cards to be fitting so that the user interface won't be so messy.
US13	As a graphic designer, I want to create visually appealing designs for the dragon cards so that they enhance players' gaming experience.
US14	As a graphic designer, I want to design the virtual game board to be easily navigable so that the players can focus on strategizing their moves rather than struggling with the interface.
US15	As a player, I want to start the game clockwise so that the game follows a structural order.
US16	As a player, I want to start the game if I am the youngest so that I can begin playing without delay.
US17	As a player, I want the dragon card(the card in the middle of volcano cards) to stay face up if it shows the same animal as my current dragon's position or if it shows a dragon pirate so that I can know how many steps I can move until I finish moving my dragon.
US18	As a player, I want to be aware that if my dragon (the player's token) is still in its cave, nothing happens so that I can plan my next turn's movement.
US19	As a player, I want to adhere to the rule that there can never be more than one dragon in the same position, the latest player should stay in the old position so that the game remains fair.
US20	As a player, I want the option to uncover another dragon card immediately after moving my dragon so that I can potentially make additional moves.
US21	As a player, I want to start and end the game so that I can start my game round and terminate whenever I want.

Table 2.1: User Stories of Basic Fiery Dragon Game Functionality

B) Team-defined Extensions of Fiery Dragons Game

User Story ID	User Story
US22	As a player, I want the game to provide a "help" button so that I can check the game rules.
US23	As a player, I want the game to show the winner with visual effects so that I can feel the excitement of victory and celebrate the win with other players.
US24	As a player, I want the game to have background music so that I can immerse myself fully in the gameplay and be entertained.

Table 2.2: User Stories of Team-defined Extensions of Fiery Dragons Game

In addition, the following link is the user stories spreadsheet that is presented in more detail and organised.

User stories Link:
User Story

3. Domain Model

A domain model is a tool primarily utilised in an agile team to define and illustrate entities and their interrelationships, collectively representing the domain space of a given problem [3]. This provides a foundation for understanding and designing systems, facilitating team maintenance, testing, and development improvement. Additionally, it establishes a significant connection between the problem domain and its implementation [3].

Therefore, we will design a domain model diagram for the Fiery Dragons game, incorporating key conceptual entities and their relationships, along with justifications in this section.

3.1 Domain Model Diagram

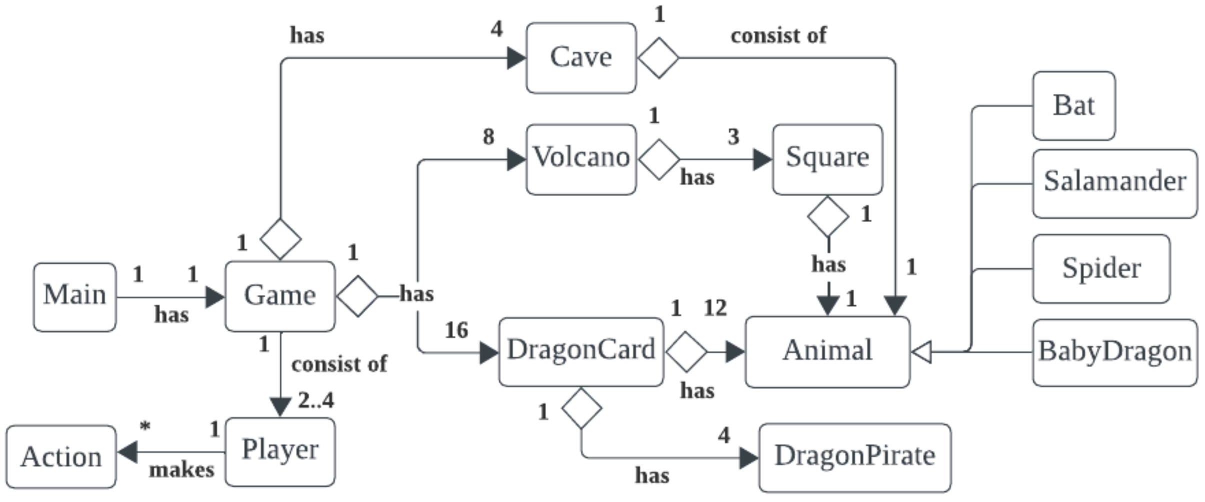


Figure 3.1.1 Domain Model Diagram of Fiery Dragons Game

3.2 Justifications

Domains	**Domain Descriptions**	**Relationships/Multiplicities Descriptions**
Main	This domain represents the entry point of the game application. Main operates and supports the game application.	**Main - Game:** It has a one-to-one relationship with the Game, representing that a round of game is created. (Association)
Game	The core game logic and it has to manage the game flow.	**Game - Volcano:** It has a one-to-many relationship with the Volcano, indicating that each game instance has 8 volcano cards. (Aggregation) **Game - DragonCard:** It has a one-to-many relationship with the DragonCard, meaning that each game instance manages 16 dragon cards. (Aggregation) **Game - Player:** The Game class has a one-to-many relationship with the Player, indicating that each game can have 2 to 4 players. (Association) **Game - Cave:** It has a one-to-many relationship with the Cave, meaning that each game instance has 4 caves. (Aggregation)
Player	The players engaged in this game. Each round of the game contains 2 to 4 players.	**Player - Action:** Player has a one-to-many relationship with the Action, indicating that a player can perform multiple actions during the game. (Association)
Action	The actions that a player can take during their turn, i.e. uncovering a	No outgoing relationship

	dragon card, moving their dragon either forward or backward, ending his/her turn, and continuing to flip the dragon card.	
Cave	The starting point for each players' dragon.	**Cave - Animal:** Each cave has only one type of Animal associated with it, and this animal type is unique for each cave. (Aggregation)
Volcano	The volcano board in which the game is played, total of 8 volcano cards, each card containing 3 squares.	**Volcano - Square:** Volcano class has a one-to-many relationship with the Square class, indicating that one volcano card consists of 3 squares. (Aggregation)
DragonCard	This domain represents the dragon cards used in the game. It can be flipped by the player in each turn. Each card contains one type of animal (from 1 to 3) indicating the steps the player should move.	**DragonCard - Animal:** Each dragon card has one type of Animal. (Aggregation) **DragonCard - DragonPirate:** There will be 4 cards containing 1 to 2 of DragonPirate. (Aggregation)
Square	The individual square on the volcano board. Each square containing one type of animal indicates that the player should uncover the dragon card which contains the same animal as the square where they stand in order to move their position.	**Square - Animal:** Each Square can be associated with one type of Animal or DragonPirate. Each animal has a movement range of 1 to 3 steps forward, while a DragonPirate has a movement range of 1 to 2 steps backward. (Aggregation)
Animal	This represents the different animals in the game i.e. Bat, Salamander, Spider, BabyDragon, which will be shown on the 4 caves respectively, the squares where the player will stand on, and on the dragon card to be picked and uncovered every turn .	No outgoing relationship The Animal class serves as a base class for specific animal types. It does not have any direct relationships with other classes.

DragonPirate	The card which can lead to the player’s dragon to move backwards 1 or 2 square(s).	No outgoing relationship
Bat	One of the animal types is represented in the game. They are associated with specific caves and squares on the volcano board.	**Bat - Animal:** Bat is a type of Animal. It inherits common attributes and behaviours from the more general Animal class. (Generalisation)
Salamander	One of the animal types is represented in the game. They are associated with specific caves and squares on the volcano board.	**Salamander - Animal:** Salamander is a type of Animal. It inherits common attributes and behaviours from the more general Animal class. (Generalisation)
Spider	One of the animal types is represented in the game. They are associated with specific caves and squares on the volcano board.	**Spider - Animal:** Spider is a type of Animal. It inherits common attributes and behaviours from the more general Animal class. (Generalisation)
BabyDragon	One of the animal types is represented in the game. They are associated with specific caves and squares on the volcano board.	**BabyDragon - Animal:** BabyDragon is a type of Animal. It inherits common attributes and behaviours from the more general Animal class. (Generalisation)

Table 3.2.1 Domain Model Diagram of Fiery Dragons Game

Specific Choices:
While drawing the domain model diagram, choices were made to ensure that all domains have their functionality and to reduce the domains based on the user stories as much as possible in order to decrease the complexity of the domain model diagram. For example, the initial decision was to have an Actor domain handle the Player and Dragon Pirate domains. However, the Dragon Pirate only affects the movement that the player can make, making it more akin to the Animal domain. Yet, based on the game rules, the Animal only has 4 types, therefore, we let the Dragon Pirate have its domain. The Action domain handles all the executable actions for the Player domain, including flipping the Dragon Card, moving the player’s dragon token, continuing flipping the Dragon Card, or ending his/her turn. Therefore, the

Action domain does not have any outgoing relationships, which are association, aggregation/composition, and generalization, based on the domain model diagram.

Furthermore, the initial design is to make the “Square” as the smallest unit to form the game board components. Thus, 3 Squares form a Volcano Card, 1 Square forms a Cave and 1 Square forms a Dragon Card. Although this approach reuses the smallest unit to build everything, it increases the complexity of the domain model. Additionally, some cards do not share the same attributes or functionality in the game and should not be in the same domain model, because it will result in an ambiguous domain model.

Assumptions:
This game is assumed to follow the physical game’s appearance and adhere to its rules. These assumptions were made based on the game rules and information from lecture slides. Furthermore, the number of dragon cards, players, volcano cards and caves are as specified in the provided information. The order of the animals on the volcano cards is also according to the provided information. Besides, this game is expected to have 4 players instead of 2 to 4 players. The players’ age is set to be 5 to 99 and it is recommended to adhere to the game rules based on the group discussion. Last but not least, only the Player domain can use the Action domain, the rest of the domains provide the related information (dependency) which will affect the Action domain. This domain model is considered a basic model and is expected to have scalability for future additional features.

3.3 Discarded Alternative Solutions

1. Player - Dragon Association (Many-to-Many): In the game, the Dragon represents the Player; however, each Dragon does not possess any special skills or irreplaceable attributes. Therefore, it is discarded to reduce unnecessary complexity and ambiguity.
2. Game - Volcano Association (One-to-Many): The game components are physically independent of each other; hence, an association relationship was initially defined for Game and Volcano. However, considering that these components are necessary to form the game from a software perspective, this solution has been discarded.
3. Game - Dragon Card Association (One-to-Many): The game components are physically independent of each other, so initially, an association relationship was defined between Game and Dragon. However, as these components are essential for forming the game from a software perspective, this solution has been discarded.
4. Game - Cave Association (One-to-Many): The game components are physically independent of each other. Hence, an association relationship was initially defined for Game and Cave. However, since these components are necessary to form the game from a software perspective, this solution has been discarded.
5. Cave - Animal Association (One-to-One): In the real world, it makes sense that an animal lives in a cave. Therefore, an association relationship was initially defined for Cave and Animal. However, considering that the cave essentially acts as an animal type and it wouldn’t be a cave without an associated animal type from a software perspective, this solution has been discarded.
6. Square - Animal Association (One-to-One): The gameboard in the game is formed by each Square. The initial idea was for each Square to be created and allow one animal to stand on it, hence an association relationship was defined for Square and Animal. However, from a software perspective, the Square essentially functions like an animal type, leading to the dismissal of this solution.

4. Basic UI Design

A low-fidelity (low-fi) prototype drawing is an early stage of user interface (UI) design and development that proposes the initial concept for the user interface of a software project.

Hence, this section presents our team's low-fi prototype drawings, including the basic Fiery Dragons game and team-defined extensions. These drawings will be accompanied by descriptions for each drawing, covering key gameplay interactions and the chosen advanced features. We utilised an external drawing tool, Figma, to create these prototypes, which will be presented in video form and summarised in the subsections below.

4.1 Low-fi Prototype Diagrams and its Descriptions

4.1.1 Home Page

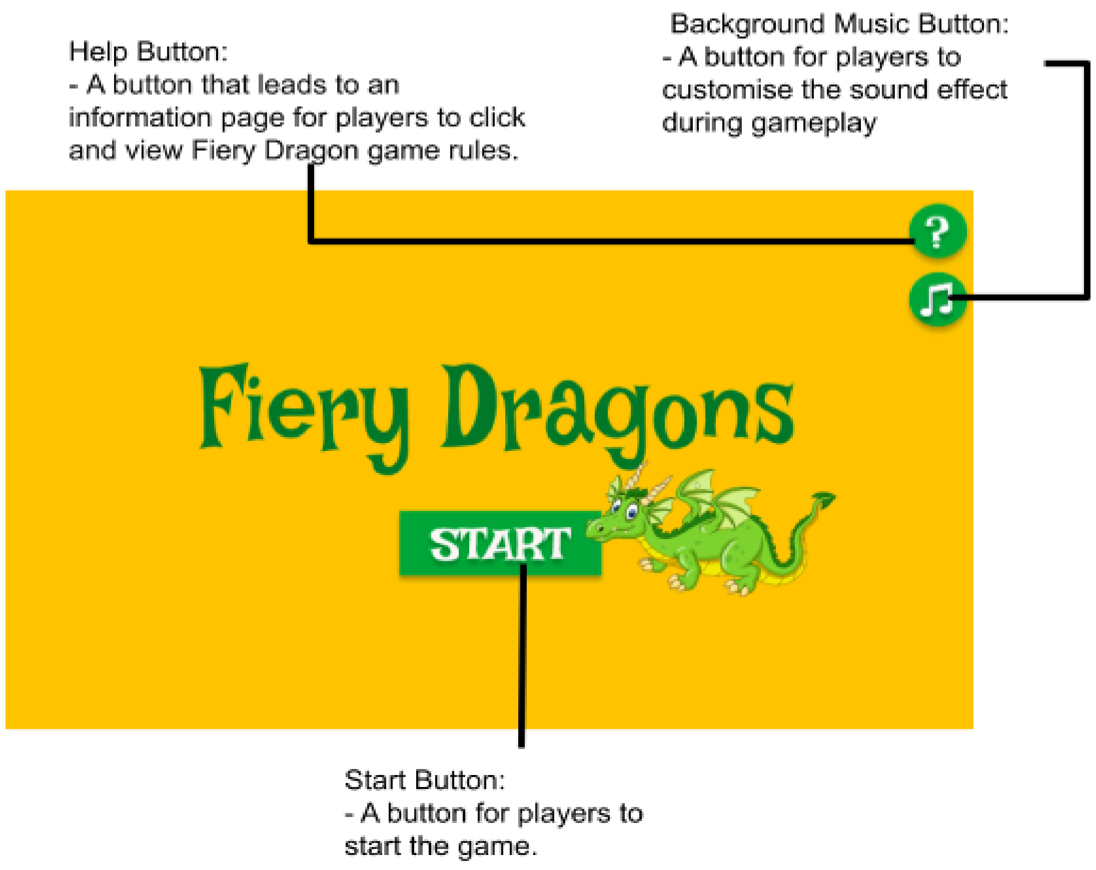
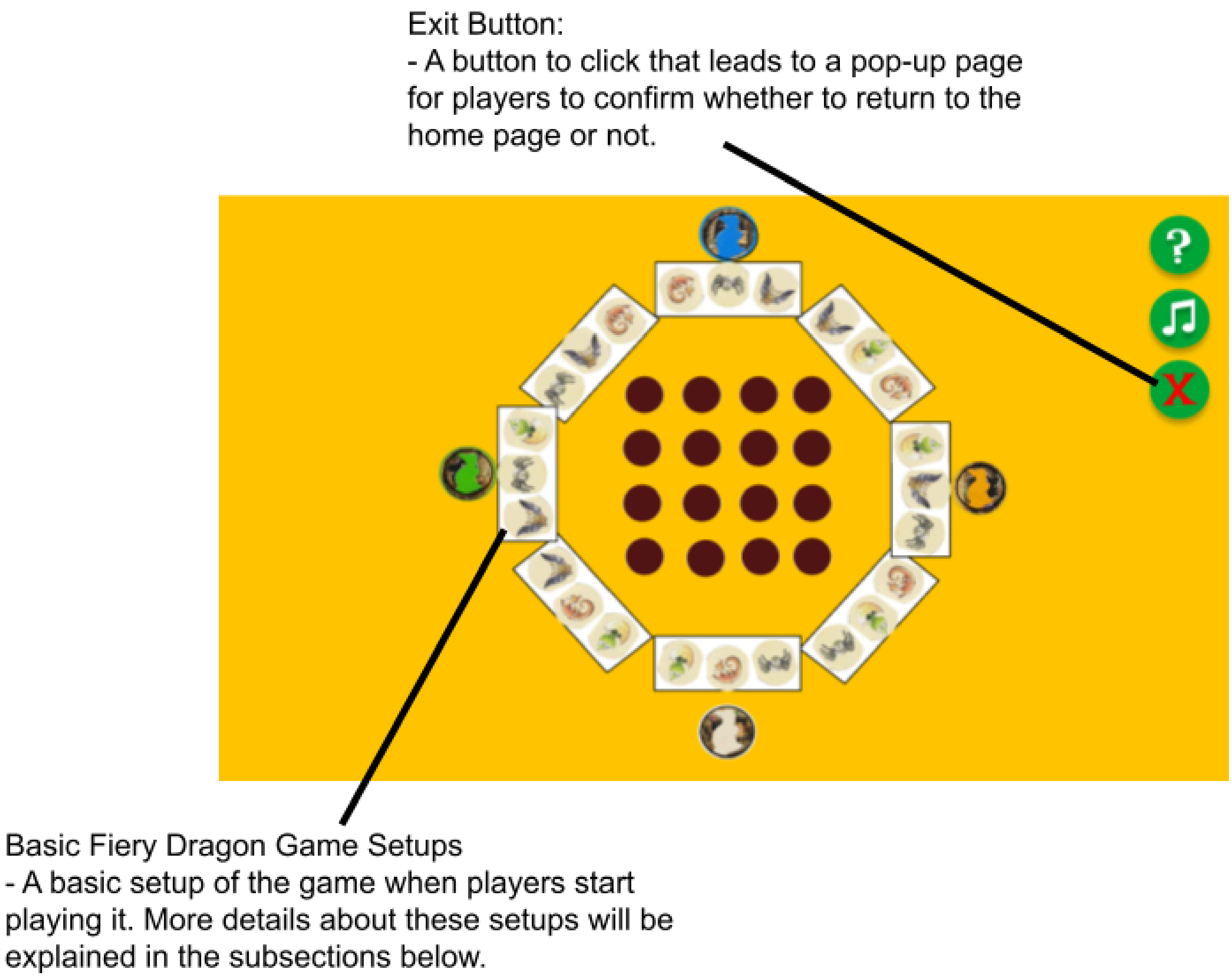

Help Button:
- A button that leads to an information page for players to click and view Fiery Dragon game rules.
Background Music Button:
- A button for players to customise the sound effect during gameplay
?
Fiery Dragons
START
Start Button:
- A button for players to start the game.

Figure 4.1.1.1: Home Page of the Fiery Dragons Game and Its Descriptions

4.1.2 Game Page



Exit Button:
- A button to click that leads to a pop-up page for players to confirm whether to return to the home page or not.

Basic Fiery Dragon Game Setups
- A basic setup of the game when players start playing it. More details about these setups will be explained in the subsections below.

Figure 4.1.2.1: Game Page of the Fiery Dragons Game and Its Descriptions

The overall game setups are shown in Figure 4.1.2.1, along with the game elements. There are 4 players, 4 caves, 8 volcano cards and 16 dragon cards shown in the game setups. Those elements will be divided and highlighted in figures as shown in the subsection below with the descriptions.

4.1.2.1 Game Elements of Fiery Dragons Game Setups

The following shows the game elements and their details of the basic Fiery Dragons game setup, as depicted in Figure 4.1.2.1:

1) **Caves**

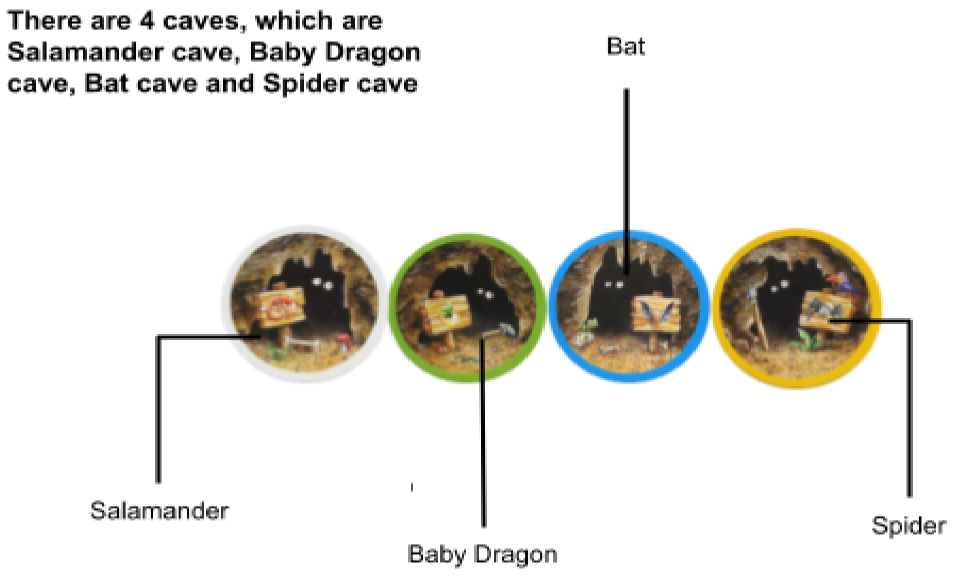

There are 4 caves, which are Salamander cave, Baby Dragon cave, Bat cave and Spider cave
Bat
Salamander
Baby Dragon
Spider

Figure 4.1.2.1.1: Caves of the Fiery Dragons Game and Its Descriptions

2) **Dragon Tokens (represent each player in different colours)**

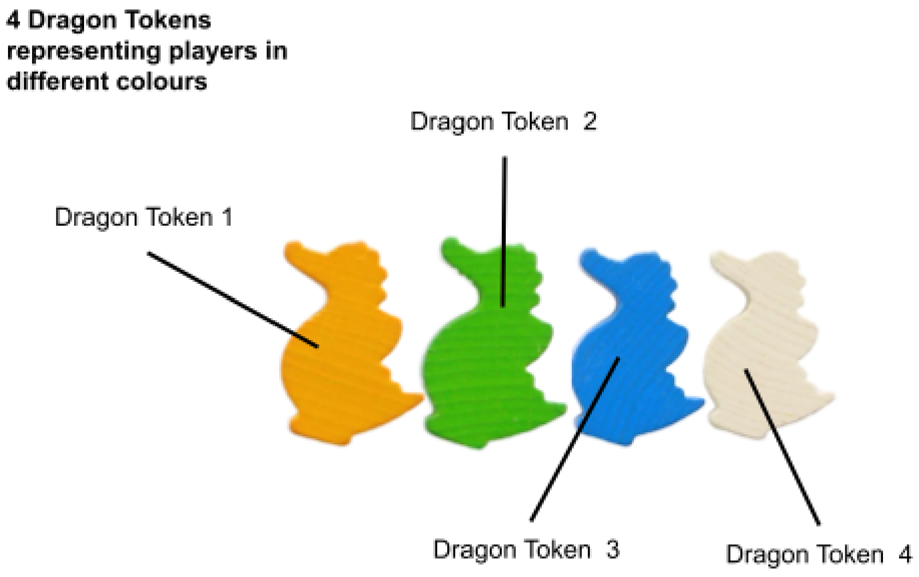


Figure 4.1.2.1.2: Dragon Tokens of the Fiery Dragons Game and Its Descriptions

3) Volcano Cards

* Note: volcano cards with cuts represent caves

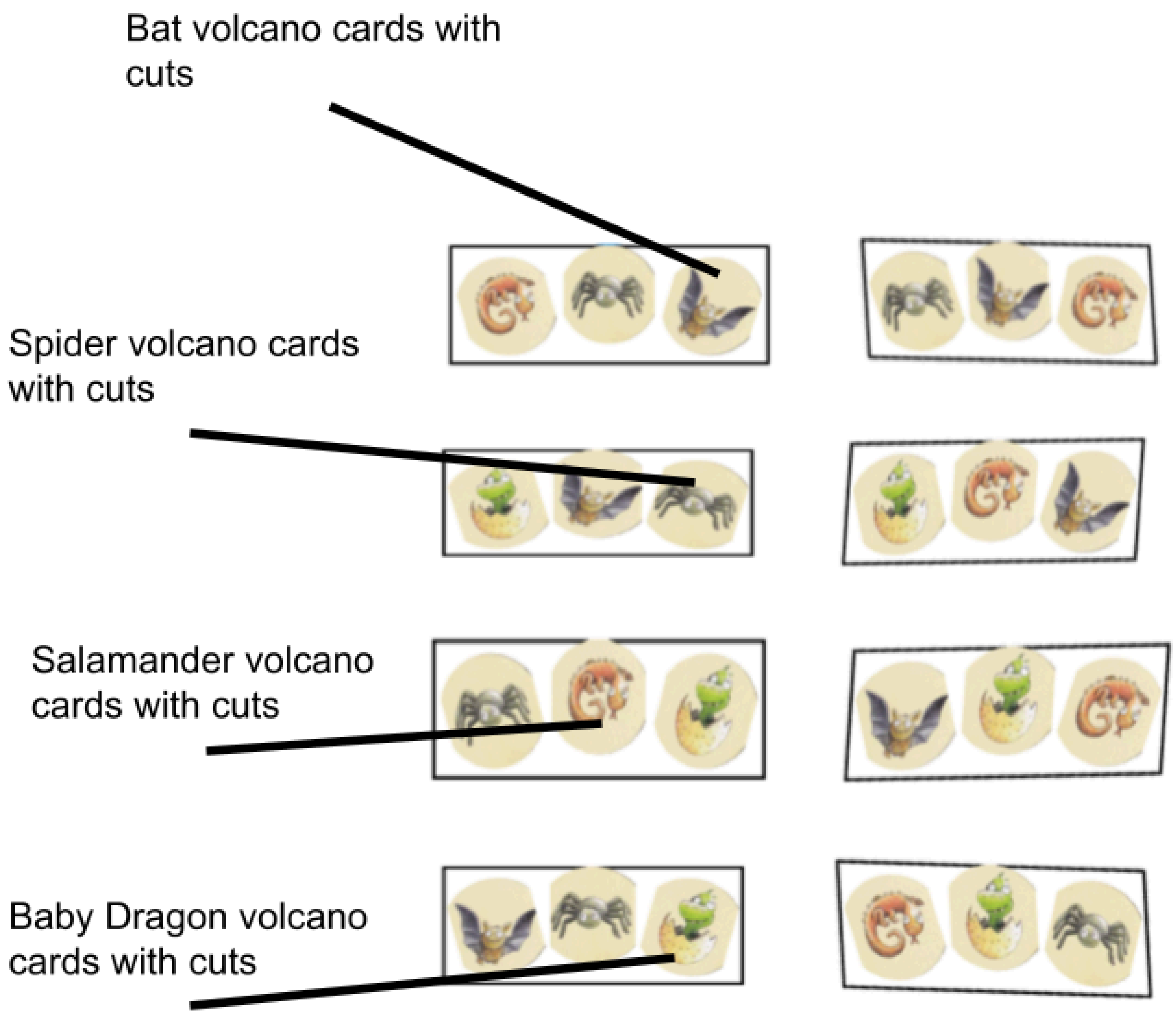

Bat volcano cards with cuts
Spider volcano cards with cuts
Salamander volcano cards with cuts
Baby Dragon volcano cards with cuts

Figure 4.1.2.1.3: Volcano cards of the Fiery Dragons Game and Its Descriptions

4) Dragon Cards

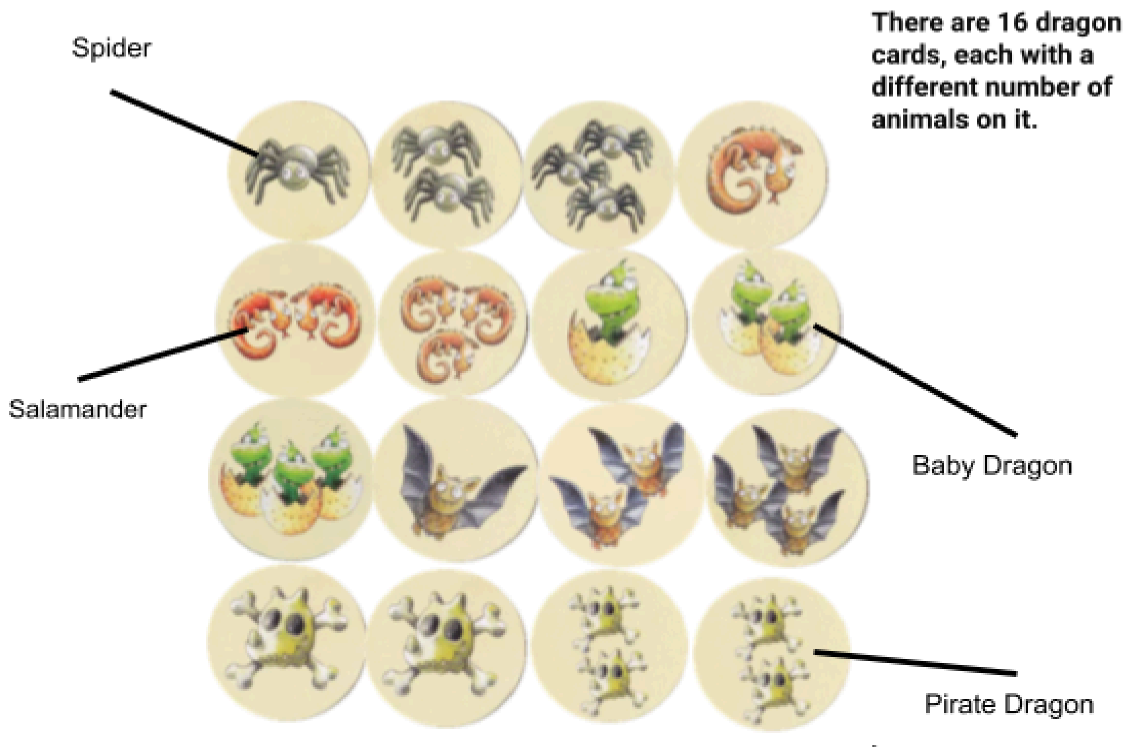

There are 16 dragon cards, each with a different number of animals on it.

Figure 4.1.2.1.4: Dragon cards of the Fiery Dragons Game and Its Descriptions

The table below shows the types of animals on dragon cards, along with each animal's numbers on each card and the respective number of cards.

Dragon Cards	Animal Numbers on the Cards	Card Numbers
Spider	1	1 (per each)
	2	
	3	
Salamander	1	1 (per each)
	2	
	3	

Bat	1	1 (per each)
	2	
	3	
Baby Dragon	1	1 (per each)
	2	
	3	
Pirate Dragon	1	2
	2	2

Table 4.1.2.1.1: Types of animals on dragon cards, animal numbers on the cards and the respective number of cards.

4.1.2.2 Overview of Fiery Dragons Game Setups

Figure 4.1.2.2 is an enlarged version of the game setups as shown in section 4.1.2 Game Page.

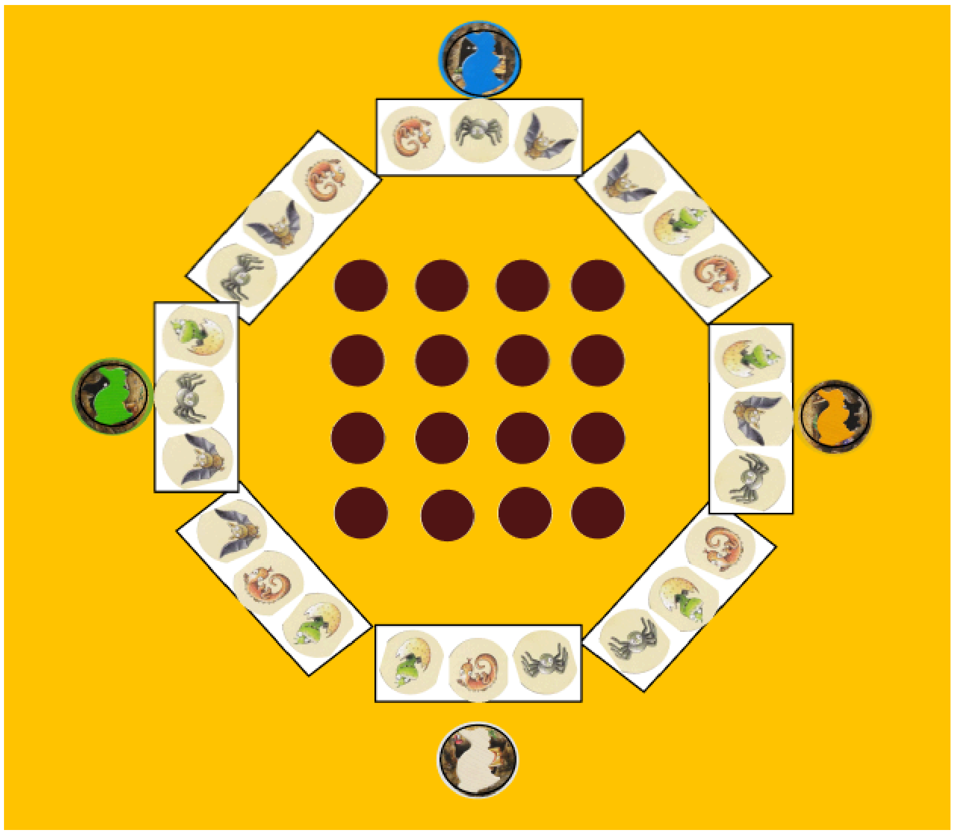
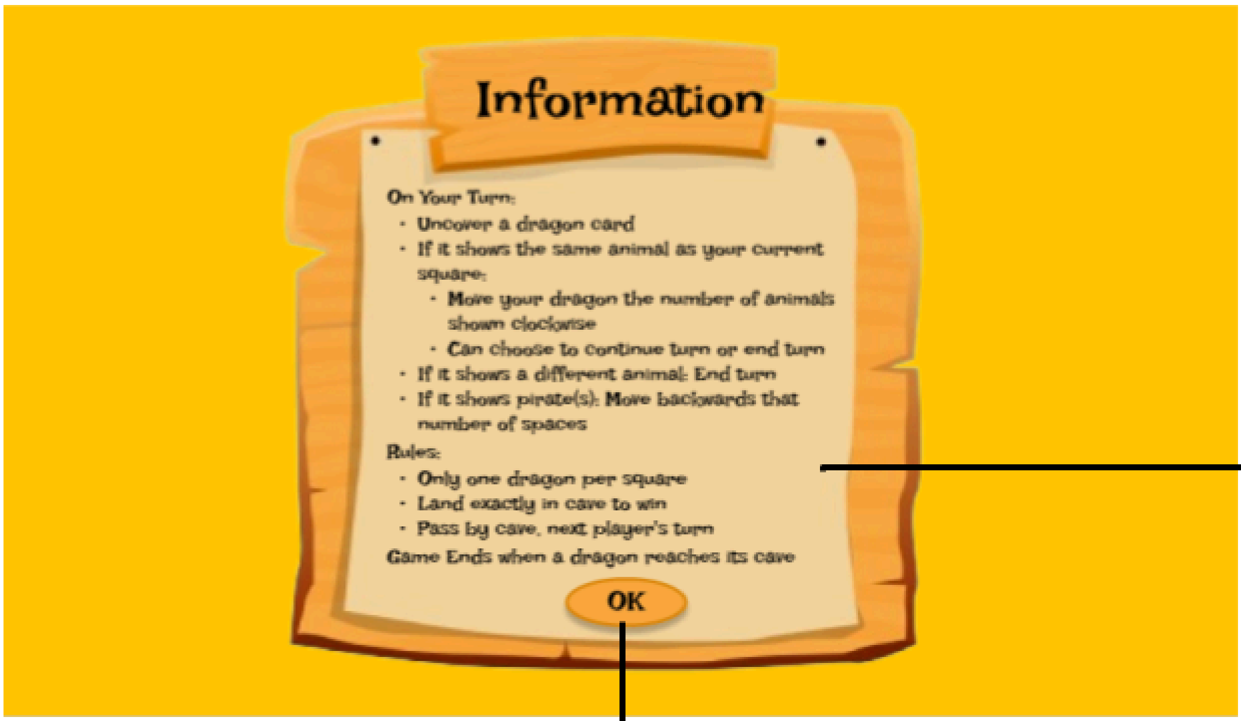


Figure 4.1.2.2.1: Overview of Fiery Dragons Game Setups

4.1.3 Information Page

Information Page:
- A page containing the game rules of the Fiery Dragons game as a guide for players.



On Your Turn:
- Uncover a dragon card
- If it shows the same animal as your current square:
  - Move your dragon the number of animals shown clockwise
  - Can choose to continue turn or end turn
- If it shows a different animal: End turn
- If it shows pirate(s): Move backwards that number of spaces

Rules:
- Only one dragon per square
- Land exactly in cave to win
- Pass by cave, next player's turn

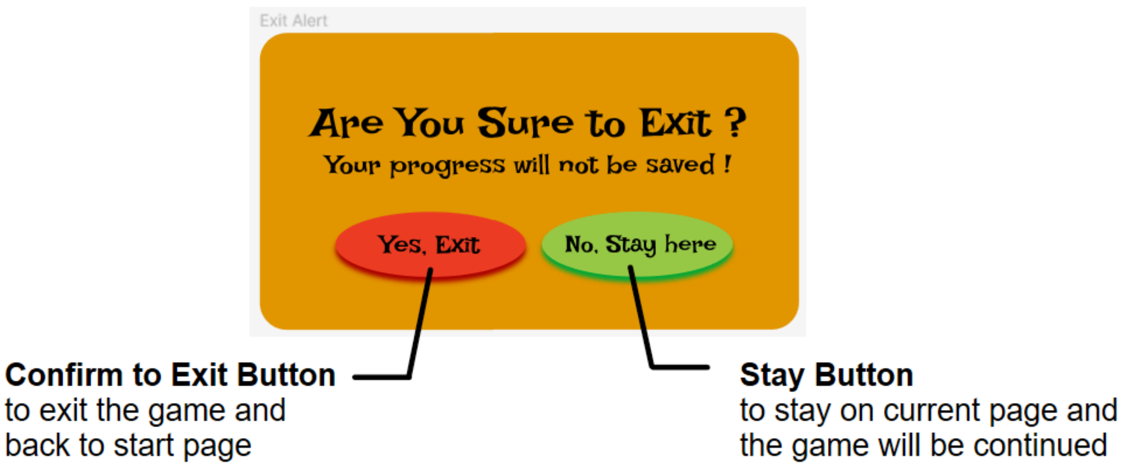
Game Ends when a dragon reaches its cave

Confirm Button
- A button for players to click and close the information page

Figure 4.1.3.1: Information Page of the Fiery Dragons Game and Its Descriptions

4.1.4 Exit Page

An Exit Page will be rendered if the player clicks on the exit button, which is shown in section 4.1.1, Home Page.


Exit Alert
Are You Sure to Exit ?
Your progress will not be saved !
Yes, Exit
No, Stay here
Confirm to Exit Button to exit the game and back to start page
Stay Button to stay on current page and the game will be continued

```

*Figure 4.1.4.1 Exit Page of the Fiery Dragons Game and Its Descriptions*
```

4.2 Key Interaction Scenarios of Fiery Dragons Game

The key interaction scenarios of Fiery Dragons Game of low-fi prototype will be covered using Figma. [Link to video demonstration]

The figures attached below are the summaries of the key interaction scenarios from the video demonstration. More detailed information can be found in our team's Figma prototype and video demonstration.

A) Scenario 1 : Starting the Fiery Dragons Game

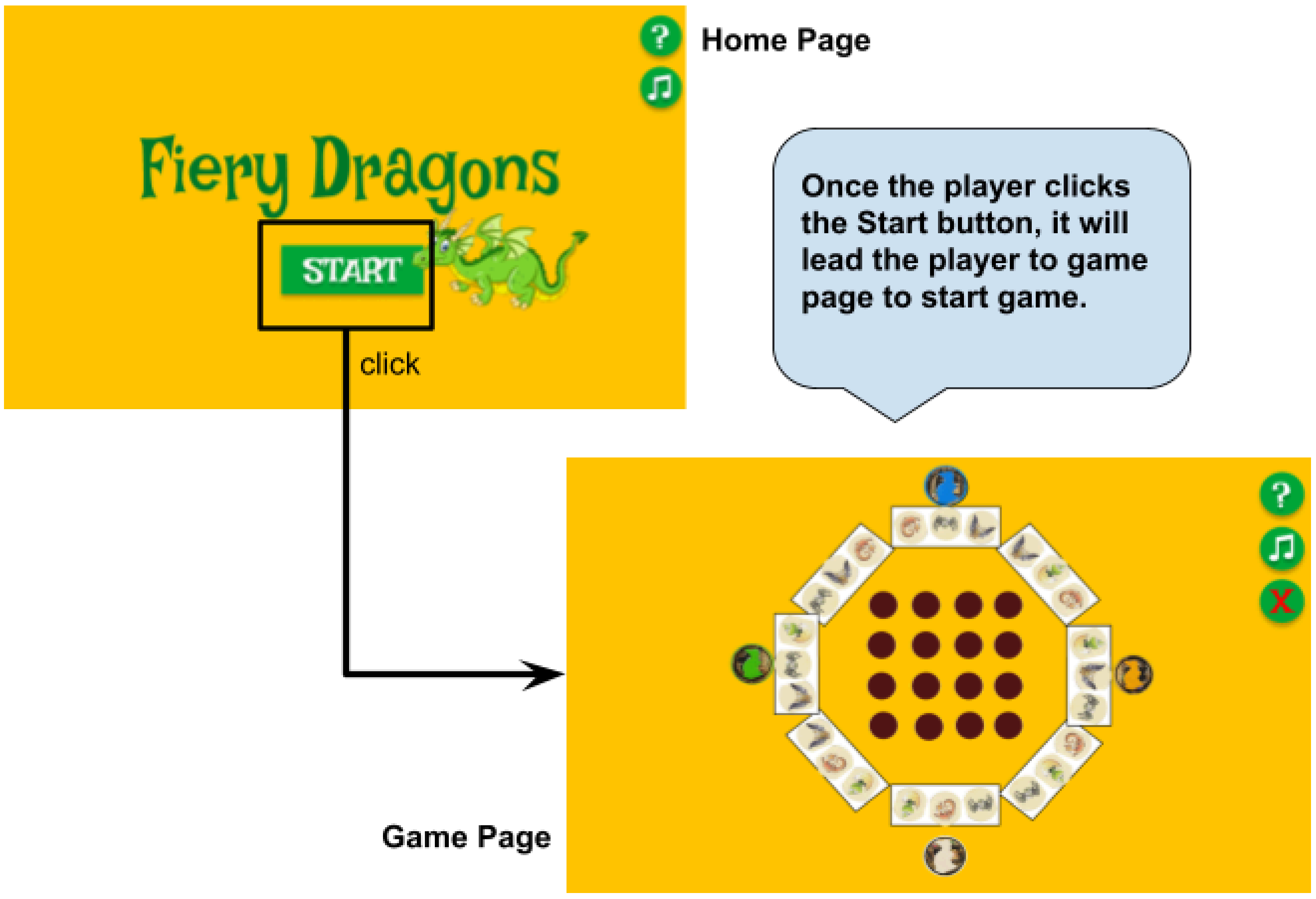

Home Page
Fiery Dragons
START
click
Once the player clicks the Start button, it will lead the player to game page to start game.
Game Page

Figure 4.2.1: Scenario when the player clicks on the Start button to begin the game

B) Scenario 2: Uncovering dragon cards of various types

When the player randomly clicks on a dragon card, as shown in Figure 4.2.2, they may uncover different possible types of dragon cards.

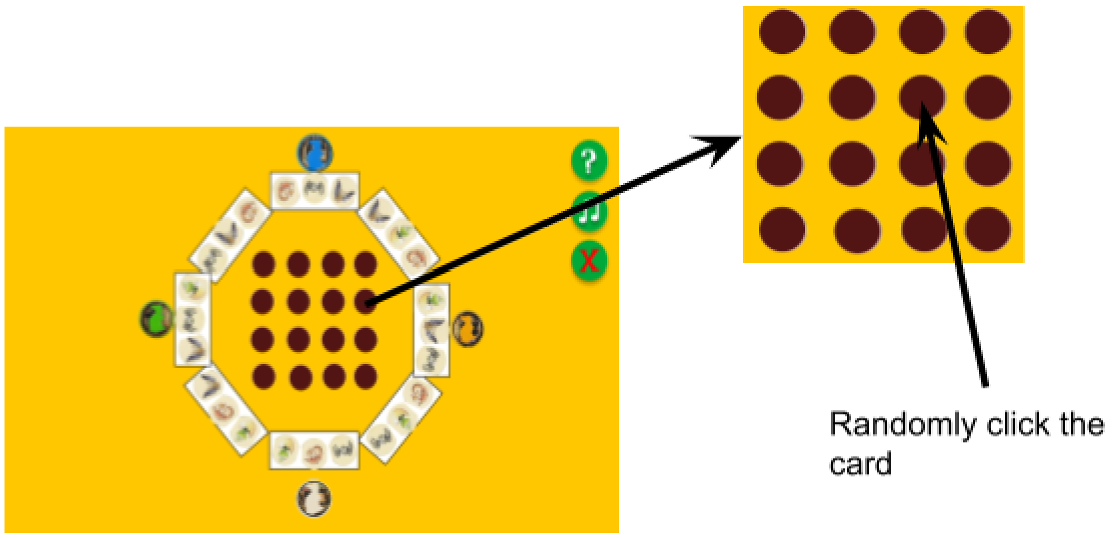


Figure 4.2.2: Scenario where the player clicks and uncovers the chosen dragon card

There are some possible types of dragon cards after randomly clicking on the card, as shown in Figures 4.2.3 to 4.2.5 below:

1) 2*Spider Card

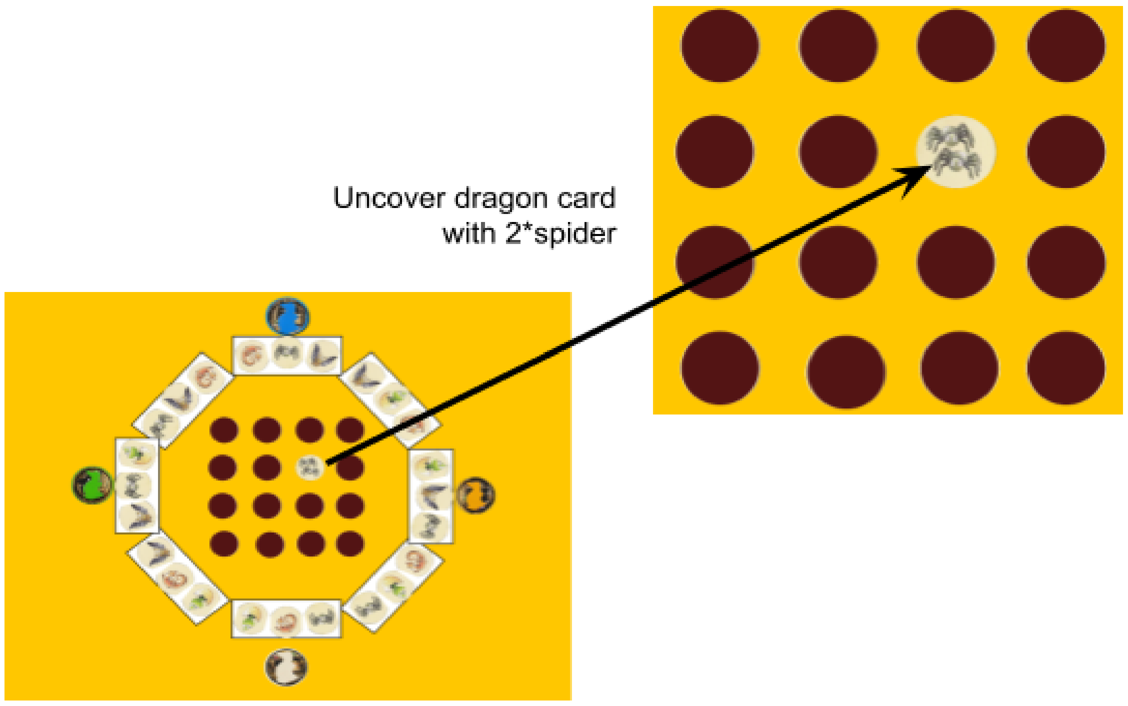


*Figure 4.2.3: Scenario where the player uncovers 2 * spider dragon card*

2) 2*Baby Dragon Card

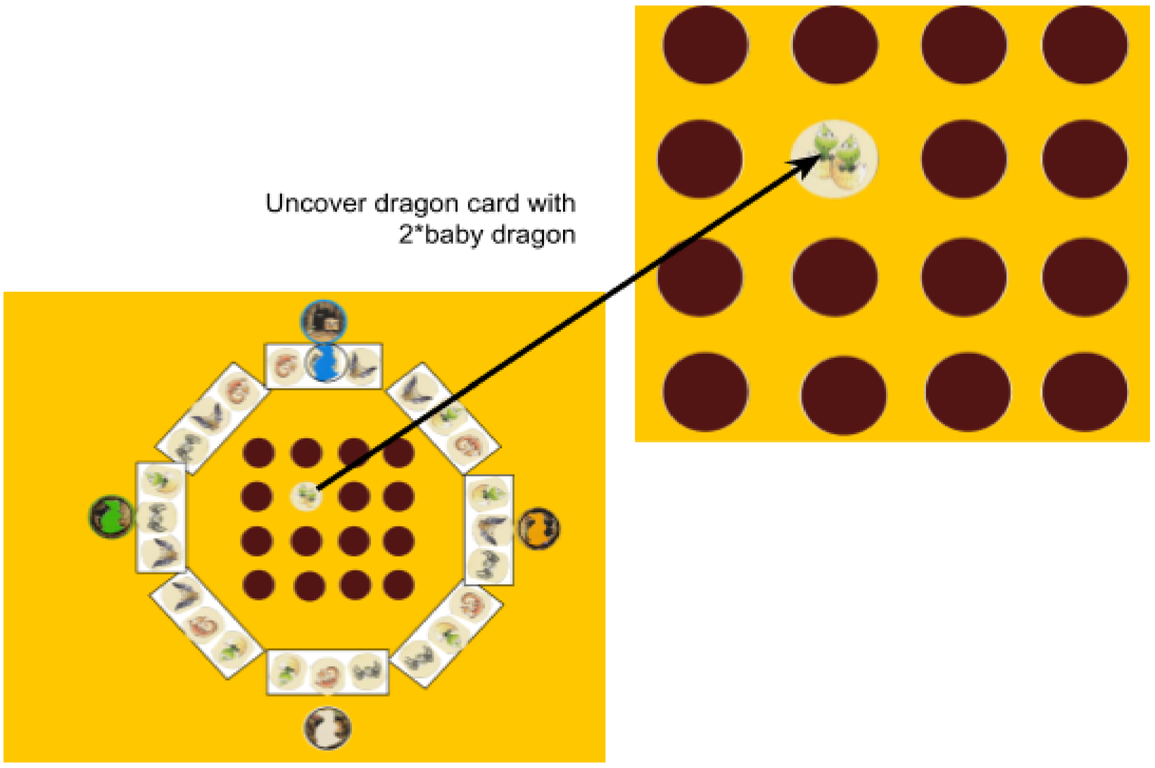


Figure 4.2.4: Scenario where the player uncovers 2 baby dragon dragon card*

3) 2*Pirate Dragon

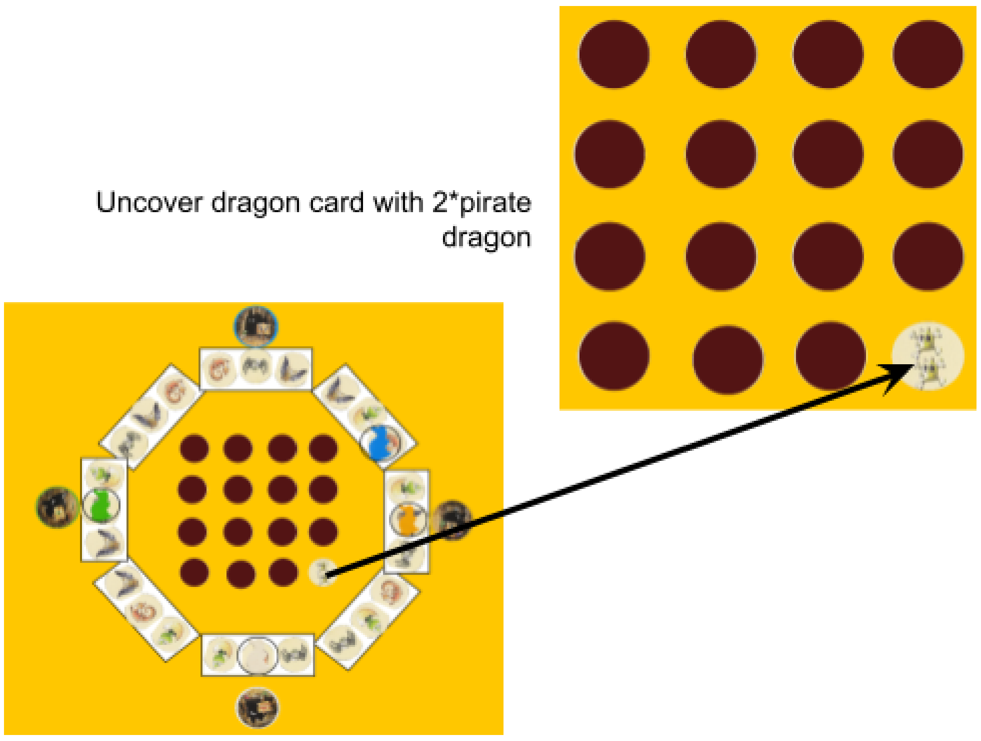


Figure 4.2.5: Scenario where the player uncovers 2 pirate dragon dragon card*

C) Scenario 3: Moving of dragon tokens

Scenario 3.1: Moving the blue dragon token when the player flips a spider dragon card matching the spider cave

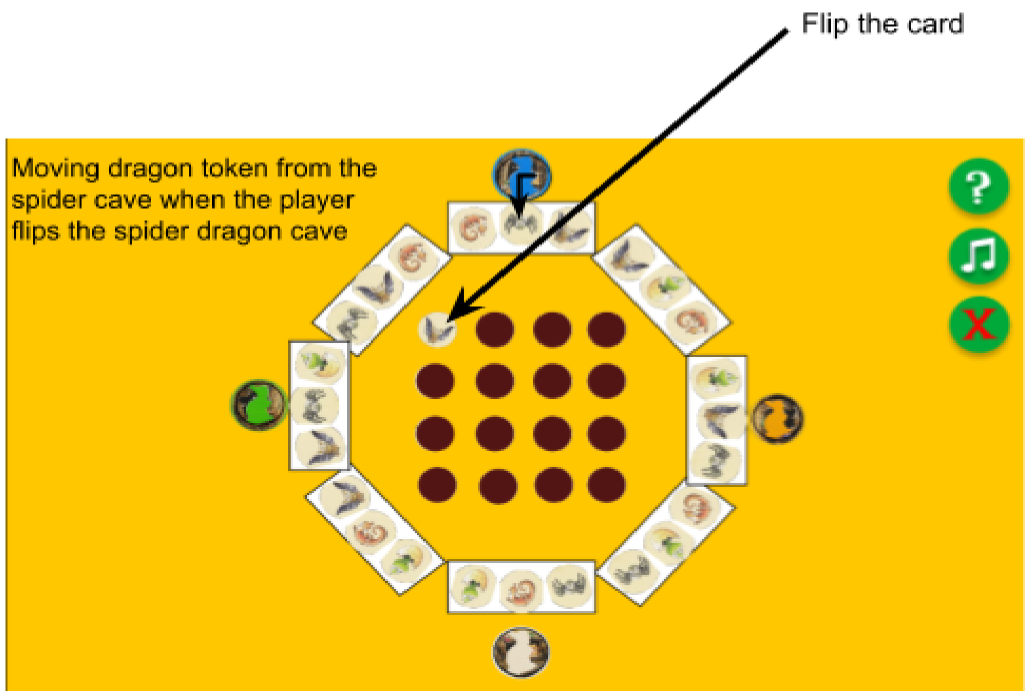

Flip the card
Moving dragon token from the spider cave when the player flips the spider dragon cave

Figure 4.2.6: Scenario where the player moves blue dragon token after flipping the spider dragon card

Scenario 3.2: The turn is over when the player flips a baby dragon card that does not match the spider cave

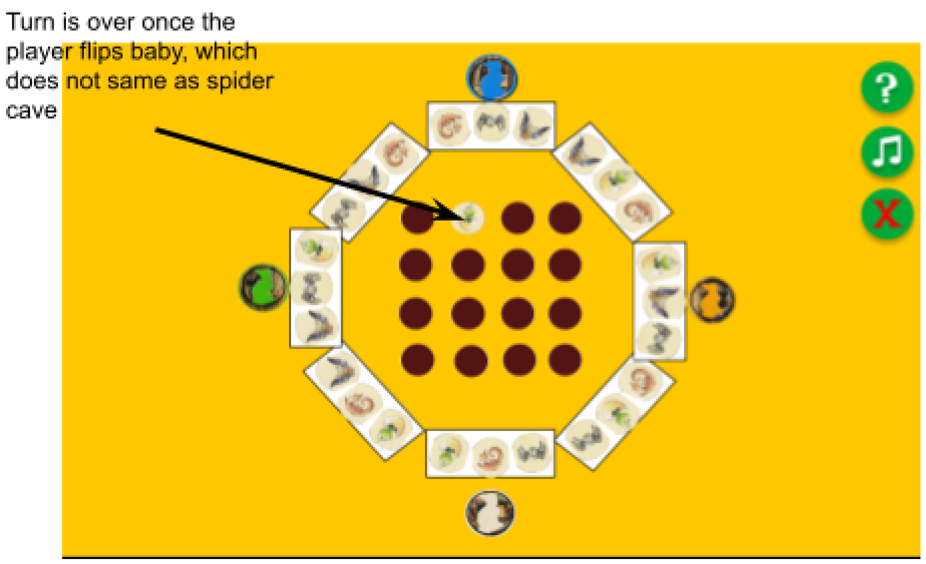

Turn is over once the player flips baby, which does not same as spider cave

Figure 4.2.7: Scenario where the player's turn is over after flipping the baby dragon dragon card

D) Scenario 4: Winning Situation

Scenario 4.1: The player wins the game by flipping 2*salamander dragon cards, and the 'Congratulations' page appears.

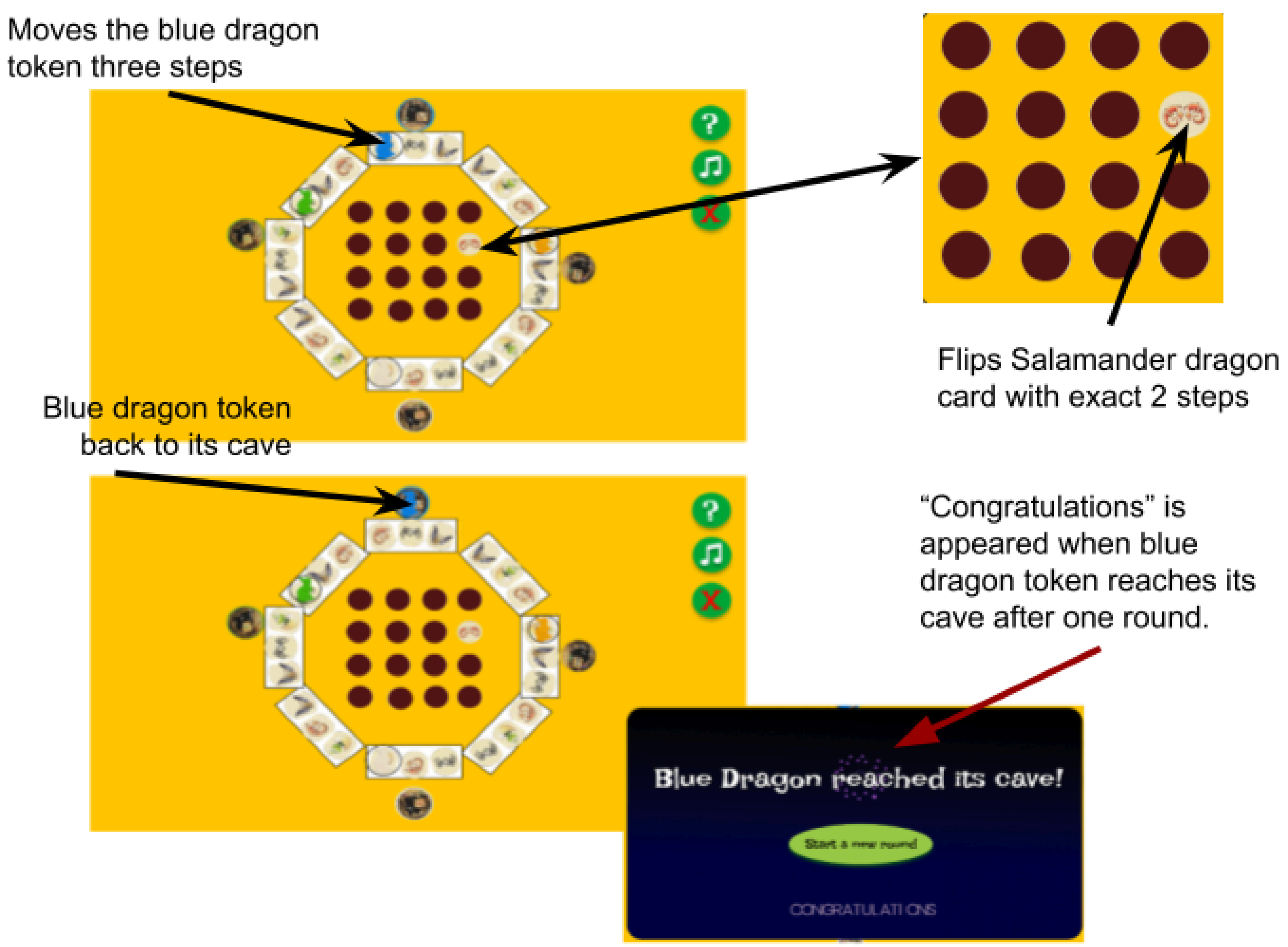
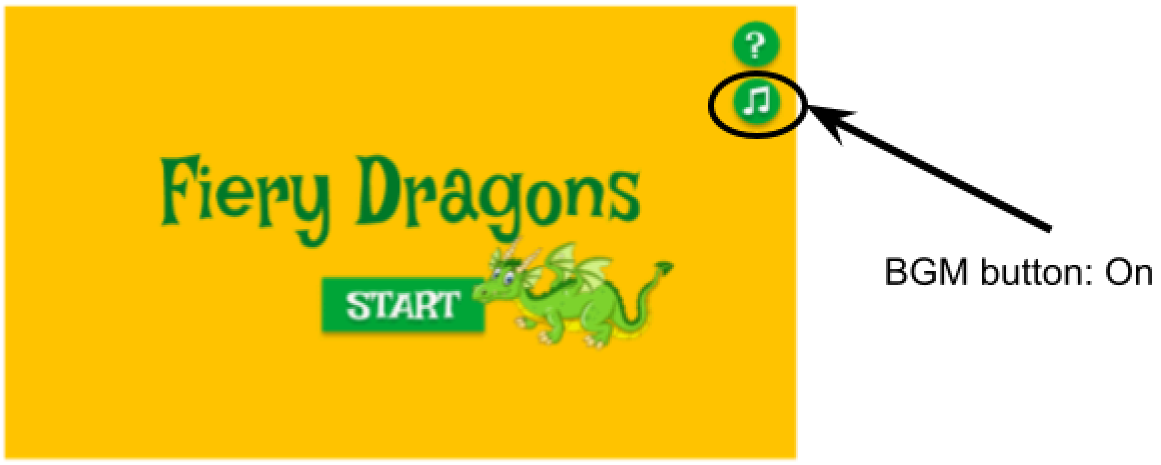

Moves the blue dragon token three steps
Flips Salamander dragon card with exact 2 steps
Blue dragon token back to its cave
"Congratulations" is appeared when blue dragon token reaches its cave after one round.
Blue Dragon reached its cave!
Start a new round
CONGRATULATIONS

Figure 4.2.8 Scenario where the player wins the Fiery Dragons Game after blue dragon token back to its cave.

E) Scenario 4: Disabled background music



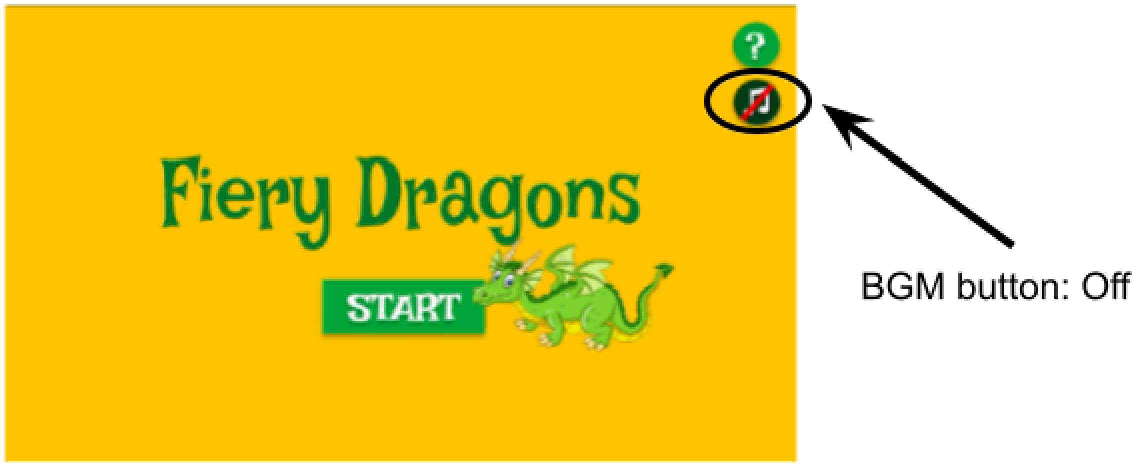
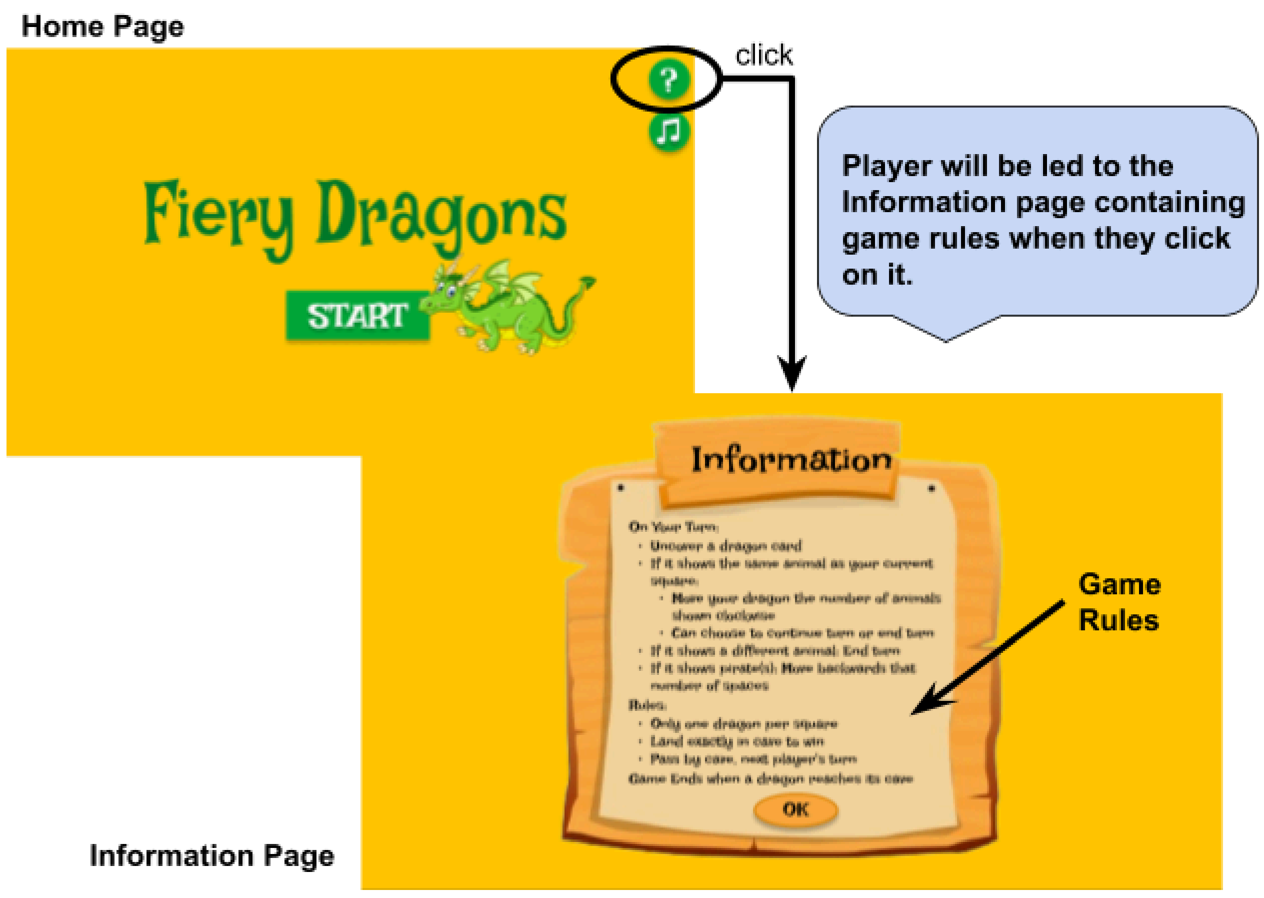


Figure 4.2.9: Scenario where the Background Music button is off to disable game's BGM

F) Scenario 5: Viewing the game rules in the Information page when the player clicks on the Help button.


Home Page
click
Fiery Dragons
START
Player will be led to the Information page containing game rules when they click on it.
Information
OK
Game Rules
Information Page

On Your Turn:
- Uncover a dragon card
- If it shows the same animal as your current square:
  - Move your dragon the number of animals shown clockwise
  - Can choose to continue turn or end turn
- If it shows a different animal: End turn
- If it shows pirate(s): Move backwards that number of spaces

Rules:
- Only one dragon per square
- Land exactly in cave to win
- Pass by cave, next player's turn

Game Ends when a dragon reaches its cave

Figure 4.2.10: Scenario where the player click Help button to view game rules

G) Scenario 6: Exiting the game when the player clicks the exit button

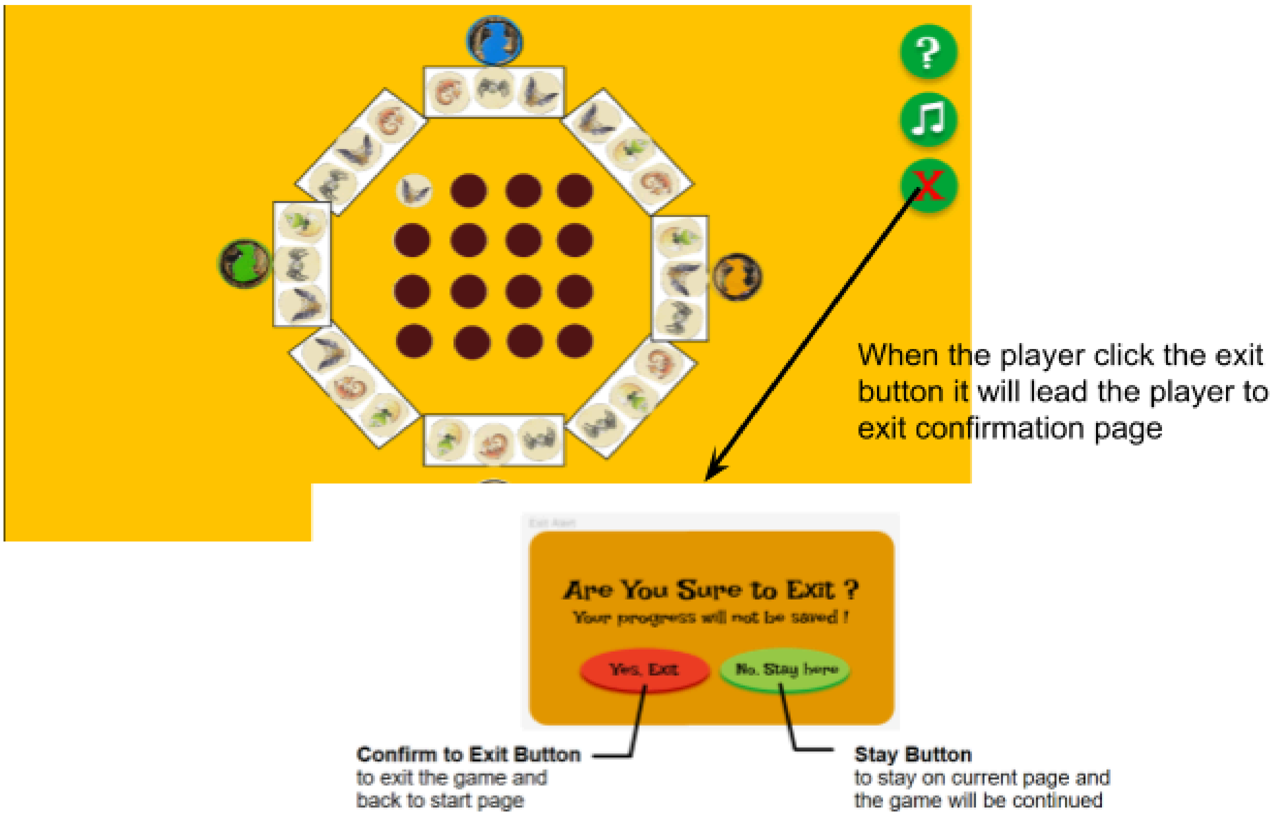

When the player click the exit button it will lead the player to exit confirmation page
Exit Alert
Are You Sure to Exit ?
Your progress will not be saved !
Yes, Exit
No, Stay here
Confirm to Exit Button to exit the game and back to start page
Stay Button to stay on current page and the game will be continued

Figure 4.2.11: Scenario where the player clicks Exit button to exit the game

4. 3 Additional Resource - Game Rules

The following are the game rules for Fiery Dragons game:

Player

- Allows 2 to 4 players.
- Ages from 5 to 99.

Game Components:

- 4 dragons with different colours represent the players.
- 4 caves (one each for Salamander, Bat, Spider and Baby Dragon)
- 8 volcano cards (4 with cuts, 4 without cuts).
    - Baby Dragon - Bat - Spider*
    - Salamander - Spider - Bat*
    - Spider - Salamander* - Baby Dragon
    - Bat - Spider - Baby Dragon*
    - Spider - Bat - Salamander
    - Baby Dragon - Salamander - Bat
    - Bat - Baby Dragon - Salamander
    - Salamander - Baby Dragon - Spider

      * is an indentation for a cave
- 16 dragon cards: (number of animals * number of cards)
    - Salamander cards (1 *1, 2 *1, 3 *1)
    - Bat (1 *1, 2 *1, 3 *1)
    - Spider (1 *1, 2 *1, 3 *1)
    - Baby Dragon (1 *1, 2 *1, 3 *1)
    - Pirate Dragon (1 *2, 2 *2)

Board Setup:

- Arrange the 8 volcano cards alternately, with cuts on every other card.
- Combine the 4 volcano cards without cuts with the cut cards to form a circular layout.
- Position the 4 caves at the cuts of any volcano card, representing the starting points for the dragons.

Card Contents:

- The caves and squares of the volcano cards show one of four animals: salamander, spider, baby dragon, and bat.
- Dragon cards display either:
    - 1, 2, or 3 animals, forward movement of the player's dragon.
    - 1 or 2 dragon pirates, backward movement of the player's dragon.

Procedure:

1. 16 dragon cards should be shuffled and face down in the inner area of the volcano cards.
2. A window should pop up to let the players key their ages. The youngest player is the first player.
3. The game will progress in a clockwise direction.
4. Start the game
    a. If the player uncovers a dragon card:
        i. If the uncovered dragon card shows the same animal as is shown on the card where the player's dragon stands:
            1. The card will stay face up.
            2. The player's dragon will move as many steps in a clockwise direction as the number of animals shown on the dragon card.
            3. The system should give the options for the player to:
                a. Continue uncovering the dragon card.
                b. End the player's turn.
            4. The player's dragon will continue moving if the player uncovers the same animal where the player's dragon stands.
            5. If the player uncovers a different animal:
                a. The player's dragon has to stay where it is.
                b. The player's turn is over.
    b. If the uncovered dragon card shows a dragon pirate:
        i. The player has to move his/her dragon backward according to the number of dragon pirates shown on the card.
        ii. The system should give the option for the player to:
            1. Continue uncovering the dragon card.
            2. End the player's turn.
        iii. If the player's dragon is still in its cave:
            1. Nothing happens if the player uncovers a dragon pirate card.
    c. Once the player's turn is over:
        i. Cover up all the dragon cards.

Rules:

- Only one dragon on a square. If the player's dragon lands on an occupied square, the player's dragon cannot move and his/her turn is over.

- The dragon will reach its cave if and only if the dragon card shows the exact number of moves. Otherwise, the dragon will pass by its caves and it’s the next player’s turn.

End of the game

- A dragon has gone around the volcano and reached its cave.

5. Appendices

Appendix 1: References

[1] D. Pandey. “Exploring Java GUI Development : AWT vs. Swing vs. JavaFX.” Linkedin. Accessed: Mar. 14, 2024. [Online]. Available: https://www.linkedin.com/pulse/exploring-java-gui-development-awt-vs-swing-javafx-divyansh-pandey-fzhxf

[2] Great Learning. “IntelliJ vs Eclipse: Which is better for beginners?.” Great Learning. Accessed: Mar. 14, 2024. [Online]. Available: https://www.mygreatlearning.com/blog/intellij-vs-eclipse/#:~:text=IntelliJ%20and%20Eclipse%20are%20both.

[3] Scaled Agile, Inc. “Advanced topic - Domain Modeling.” SAFe STUDIO. Accessed: Mar. 16, 2024. [Online]. Available: https://scaledagileframework.com/domain-modeling.

Appendix 2: Important Links

This section will include all the important links, user stories, GitLab repository, contribution logs, wiki section in GitLab repository and Google Drive that facilitate our team's collaboration and understanding of users' requirements more effectively. It also includes links to low-fidelity (low-fi) prototype and also video demonstration of gameplay interactions using Figma.

Google Drive Link:

FIT3077-Submission

User Stories Link:

User Story

GitLab Repository Link:

https://git.infotech.monash.edu/FIT3077/fit3077-s1-2024/MA_Tuesday

Contribution Logs:

FIT3077 - MA_Tuesday12pm_Team002 -Contribution Logs

Wiki Section in GitLab Repository Link:

https://git.infotech.monash.edu/FIT3077/fit3077-s1-2024/MA_Tuesday12pm_Team002/-/wikis/home

Low-fidelity (low-fi) Prototype (Figma) Link :

https://www.figma.com/file/5x3PHuAQZPU4ipMkI7UW7O/Untitled?type=design&node-id=0%3A1&mode=design&t=Sja006WvEqmrT0FO-1

Video Demonstration of Gameplay Interactions using Figma Link:

https://youtu.be/poqolLUGaR8