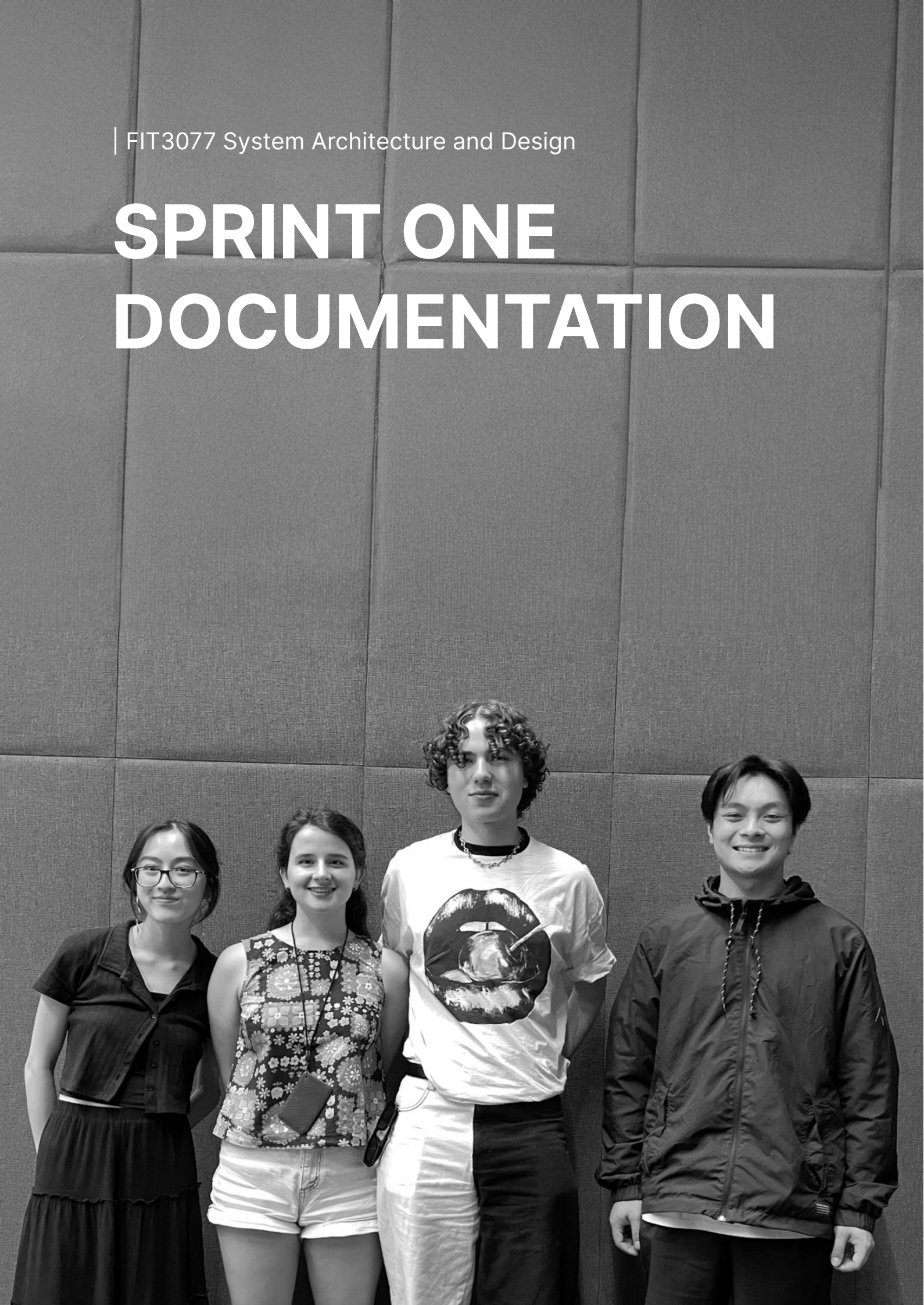
| FIT3077 System Architecture and Design

SPRINT ONE DOCUMENTATION



Team Information

Team Name

Team Code	CL_Monday06pm_Team069
Name	“There’s nothing to see here”

Team Members

Audrey Phommasone

Contact	apho0008@student.monash.edu
Technical Skills	Python • Java • UI Design • Leadership • Teamwork
Fun Fact	I have started learning 7 different languages

Tye Samuels

Contact	tsam0016@student.monash.edu
Technical Skills	Python • Java • Matlab • CSS • HTML • UI Design Graphic Design • Teamwork • Leadership Communication • ASCII Artist
Fun Fact	I turn on “Show non-printing characters” on docs so I can remove trailing spaces

Jun Hao Ng

Contact	jngg0122@student.monash.edu
Technical Skills	Python • Java • PHP • Unit Testing • Validation
Fun Fact	I play state level volleyball

Georgia Kanellis

Contact	gkan0011@student.monash.edu
Technical Skills	Python • Java • C++ • Matlab • Javascript • CSS • HTML Teamwork • Leadership • Communication
Fun Fact	I have 112% completion in Hollow Knight

Team Schedule

Our team is dedicated to regular, weekly meetings in order to stay updated and on top of the project. Our team meets on Tuesday from 12:00 to 14:00 in person, as well as online on Saturday from 9:00 to 10:00. These meetings are in addition to our weekly FIT3077 workshops where we also discuss and touch base.

It is important to have a clear understanding of the requirements before allocating tasks. The weekly meeting will serve as both a check-in and an opportunity to consolidate the requirements. Workload will be distributed through team discussions and managed depending on availability of individual group members. Additionally, it is important that each team member has the opportunity to contribute to all aspects of the project.

Technology Stack and Justification

Our team has decided to use Java as our programming language for our project, as it is better suited to Object-Oriented programming than Python, allowing us to structure our project in a more organised manner. This allows us to access more technical and niche functionality that python cannot offer as elegantly. This also aligns with the team's skill sets as many of us have worked extensively in Java previously. Each team member has their own preference of IDE but to streamline and simplify the work, all members of the team will be using IntelliJ IDEA. All of our files will be shared and worked on collaboratively through Gitlab using CI/CD.

The APIs we will be using to help create the solution is Swing. This will allow us to create an executable which will open a new window where the game can be played interactively. The required compatibility of the game is maintained by Swing's lightweight and platform independent implementation, with a comparatively faster execution time than other Java APIs, such as AWT [1]. Swing includes support for HTML, with limited inbuilt styling capabilities, which will be sufficient in enabling us to display and update the game UI.

The team is quite familiar and self-sufficient enough with all the technologies and APIs above to require minimal support from the teaching team. The only support the team may seek is information about the project requirements and its scope and/or approval for the usage of any additional technologies or project extensions that may be found during the development of the solution.

When discussing which programming language to use, a spike was undertaken to demonstrate the capabilities of Java in implementing the desired ASCII aesthetic of our game UI in addition to exploring the functionality offered by the Swing library. The spike was successful in exploring the use of Swing components such as JFrames and JLabels. Moreover, by successfully generating ASCII art shapes, the Spike has validated our chosen game aesthetic as both achievable and extensible.

User Stories

Game Functionality / General

- As a player, I want to be able to enter my age so that the starting player can be selected.
- As a player, I want to be able to enter when I last stroked a dragon so that the starting player can be selected.
- As a player, I want the game loop to end upon satisfying the winning condition so that the game can be concluded.
- As a player, I want the result of the game (i.e the winner) to be displayed so that the game outcome is known.

Dragon Cards

- As a player, I want to be able to select a dragon card so that my character can progress in the game.
- As a player, I want to be able to see the dragon card value once I have turned it over so that I can know what I have selected.
- As a player I want to be able to see which cards are available to turn over so that I can continue to progress over the board.
- As a player, I want to be able to choose not to pick up another dragon card so that I can have more freedom to strategise within the game.
- As a player I want to be able to see which cards I have already turned over so that I may not accidentally select them again.
- As a player I want the cards I turned over to be face down at the end of my turn so that other players cannot see what animals are on those cards.

Board / Volcano Cards

- As a player, I want to be able to see the location of the caves so that I know where I am aiming to move my character towards.
- As a player, I want to be able to see the location of my character on the board so that I can see my progression in the game.
- As a player, I want to be able to see the other players' characters so that I can see my progression in comparison to my opponents.
- As a player, I want to be able to check if a game square is already occupied so that I can tell if I am able to move to that square.
- As a player, I want to be able to identify the species in the square I am currently occupying so that I know which cards to aim for.

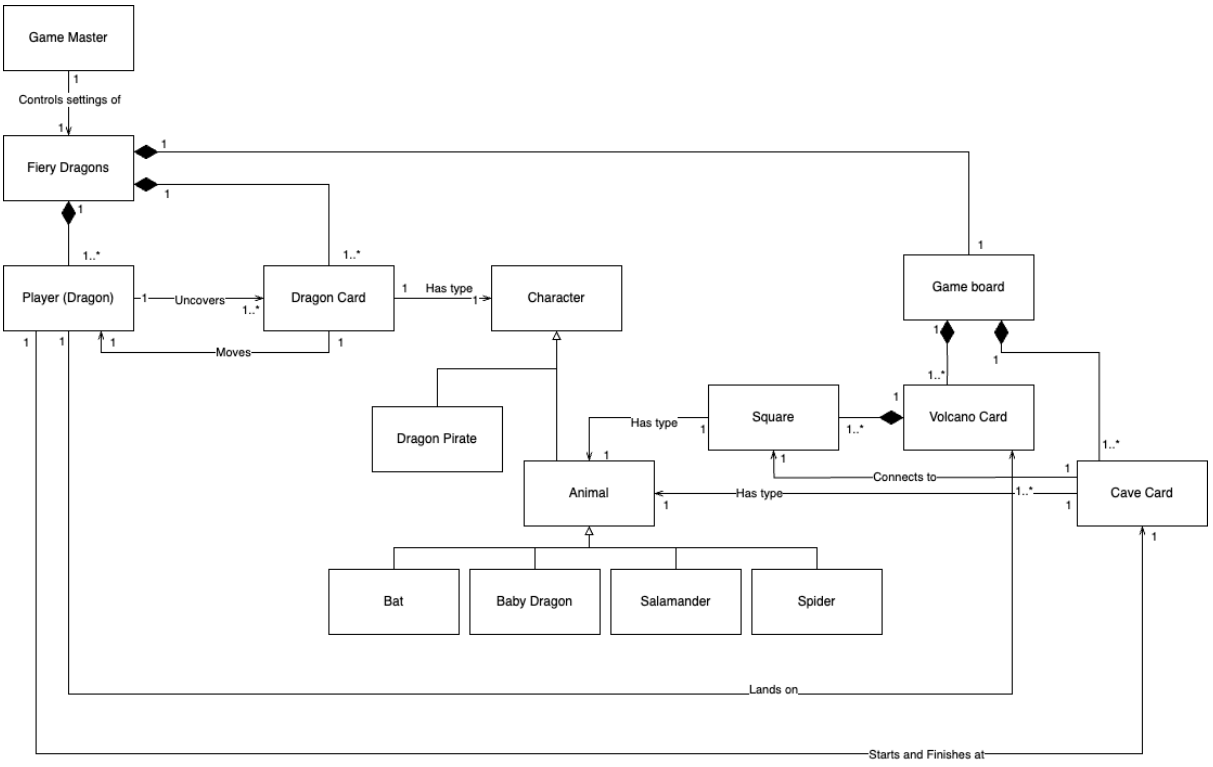
Extensions

- As a game master, I want to be able to modify the number of players in a game so that more people are able to play.
- As a game master, I want to be able to modify the effects of dragon cards so that I can increase the types of moves within the game.
- As a game master I want to be able to modify the number of volcano cards in the board so that I am able to prolong the game.
- As a gamemaster, I want to be able to modify the number of dragon cards available so that I can adjust the difficulty of the game.
- As a game master I want to be able to enable multiple extensions on the base game so that I am able to increase the quality of the game and make it more fun.
- As a game master, I want cave cards to be evenly distributed around the board regardless of the number of players/volcano cards so the active extensions will still maintain the core game mechanics.
- As a game master, I want to be able to change the dragon card mechanic so that the cards shuffle after every round so that it becomes a game of chance to add a new layer of difficulty to the game.

- As a game master, I want to be able to extend the game loop so that the remaining players can compete for second place, third place and so on.

Domain Model

Figure 1.1: Full game domain model including some extensions.



Domain Model Justification

Extensions are not included in the base game domain model, but are shown to demonstrate that it is expandable. The extensions we may include:

- Additional Volcano Cards to increase the size of the board and increase the duration of the game
- Additional Dragon Cards to lower the chances for each card and adjust the difficulty of the game
- More Types of Dragon Cards to enhance the complexity of the game
- More players to increase the competitiveness of the game

In alignment with the game rules project brief we inferred from the listed game pieces the following objects:

- **Fiery Dragons**
    - All the entities come together to form the overall Fiery Dragons Entity. Fiery Dragons, the board game, is a composite of many dragon cards and one game board, as it cannot exist without those entities. It also is an aggregate of many players. We decided to not have the maximum player number of four players, as one of our extensions involves increasing or decreasing the number of players within the game.
    - We decided not include Rounds as a related entity, despite including it in our previous iteration of the domain model. Although Rounds exist in the original game conceptually as a way of monitoring the player turn order, it was decided that rounds did not exist as a specific tangible entity that we could define relationships for. Instead, rounds were considered in the form of the iterative process of Players uncovering Dragon cards.
- **Game Master**
    - The Game Master enables specific extensions or changes the settings of the Fiery Dragons game. An association was drawn between Fiery Dragons and the Game Master rather than individual aspects of the game due to extensions having an effect on a range of separate entities. Thus to reduce additional complexity of the diagram this was abstracted to the Game Master's connection with the overall game.
- **Player (Dragon)**
    - One Player, represented as a Dragon token, uncovers one or more Dragon Cards each turn. This is because a Player may uncover multiple Dragon Cards within turn, but also may stop at any point in time after the first card.
    - The Player starts and finishes within a single cave card. The Player/Dragon entity also has another association relationship with the Volcano Card entity. This has a multiplicity of one to many, as the Player must travel over and land on multiple volcano cards throughout the game loop.

    - Although Players could be considered to be separate entities to the Dragon tokens, the domain model was drawn with these as synonymous entities as an abstraction of the game where any and all actions by the Player affect the Dragon.

- **Dragon Card**

    - The dragon card has a one to one association with the Character, as each dragon has one character type. The Dragon Card also has the ability to move the Dragon, and thus has an association relationship.

- **Character**

    - The Character is a generalisation of the types a Dragon Card can have. These include Dragon Pirate, and Animal. This entity has the ability to be expanded upon within our extensions, for example, adding more character effects.

- **Dragon Pirate**

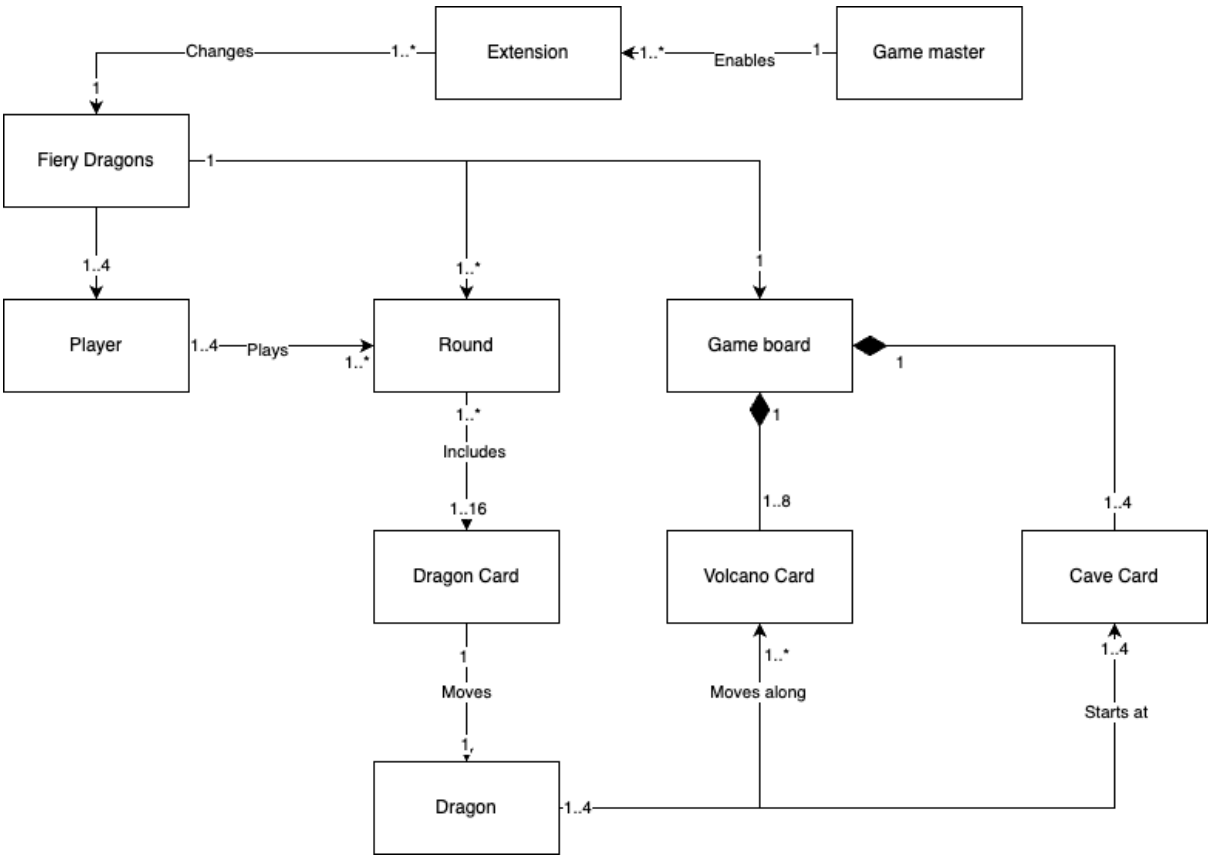
    - The Dragon Pirate specifies the character general domain entity. We decided not to include the Dragon Pirate entity as a specification of the general Animal entity due to the association relationship between Squares and Animals. Each Square has an Animal type, but these types do not include Dragon Pirate, thus it was inappropriate to have Dragon Pirate be a specific type of Animal, as it has different behaviour and game rules.

- **Animal**

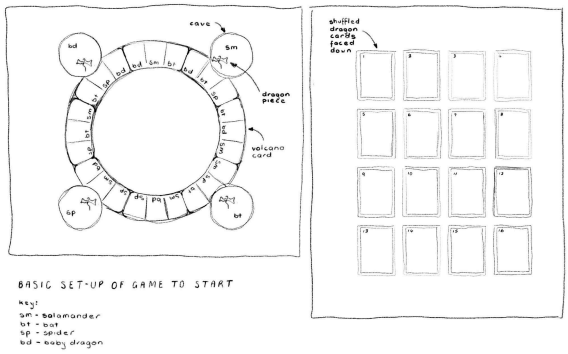
    - Animal is another entity that specifies the character domain entity; it itself is a generalisation of Animal entities, all of which match to both a Cave type and number of individual Squares on the Volcano Cards. These include Bat, Baby Dragon, Salamander and Spider. Although these could have been treated as separate entities without the generalisation, this allowed for the entities to be easily grouped, and will allow for the design to be further extended to include more animal entities as a way to expand the Dragon Card effects.

- **Square**
    - The Square entity is included as it is needed to make up a Volcano Card, and thus has a composite relationship. We have made the multiplicity of this entity to be one to many, even though the base game only would have three squares making up each Volcano Card. This decision was made to allow for the increase of squares within a Volcano Card as a game extension option.
- **Game Board**
    - As seen in the game rules, the Game Board is typically made up of eight Volcano Cards, and four Cave Cards. Within the domain model we have instead made the composite relationship multiplicities to be from one to many, as we have planned to be able to expand the Game Board, and the number of players in the future as game extension options.
- **Cave Card**
    - One Cave Card has one unique Animal type within the game, and thus has an association relationship between those two entities. As each Cave Card connects to a Square on a Volcano Card, we subsequently added a relationship between the two entities.

Figure 1.2: Previous iteration of the domain model.

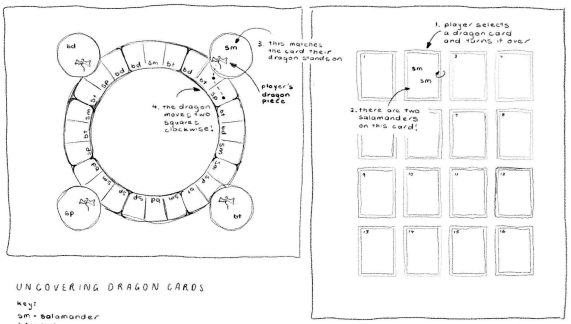


Basic UI Design



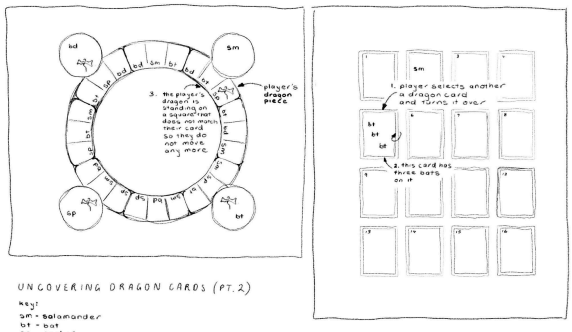
BASIC SET-UP OF GAME TO START

key:
sm - salamander
bt - bat
sp - spider
bd - baby dragon



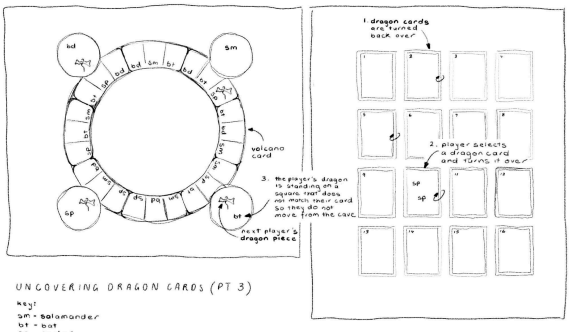
UNCOVERING DRAGON CARDS

key:
sm - salamander
bt - bat
sp - spider
bd - baby dragon



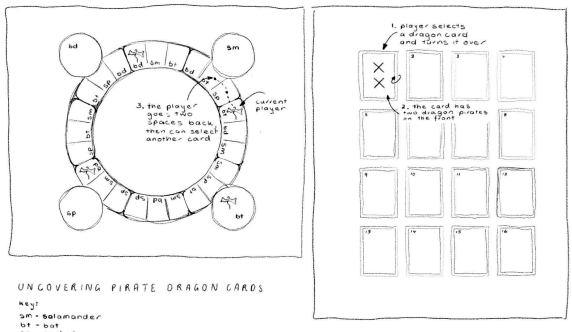
UNCOVERING DRAGON CARDS (PT. 2)

key:
sm - salamander
bt - bat
sp - spider
bd - baby dragon



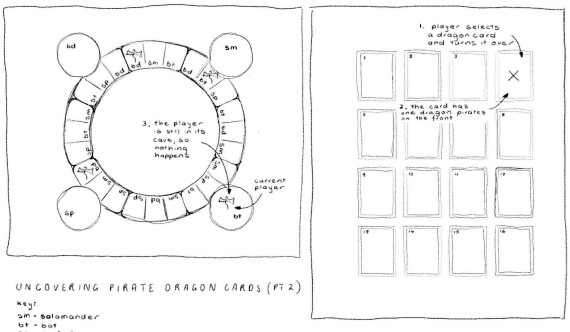
UNCOVERING DRAGON CARDS (PT 3)

key:
sm - salamander
bt - bat
sp - spider
bd - baby dragon



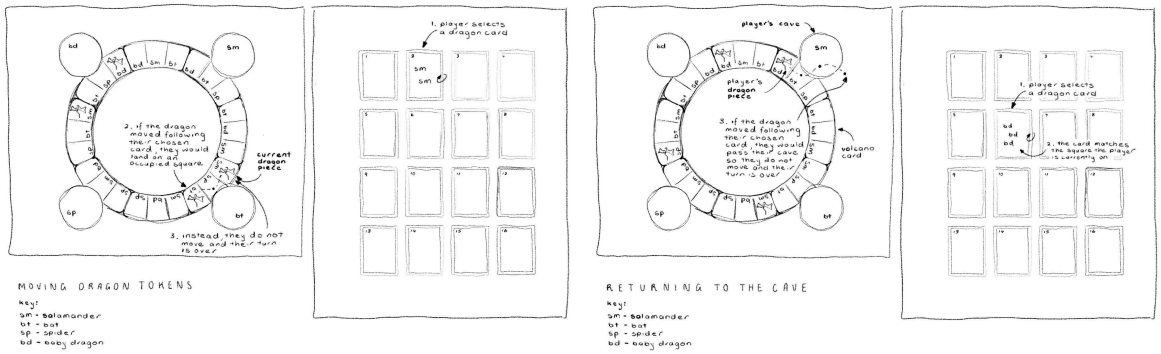
UNCOVERING PIRATE DRAGON CARDS

key:
sm - salamander
bt - bat
sp - spider
bd - baby dragon
X - pirate dragon



UNCOVERING PIRATE DRAGON CARDS (PT 2)

key:
sm - salamander
bt - bat
sp - spider
bd - baby dragon
X - pirate dragon

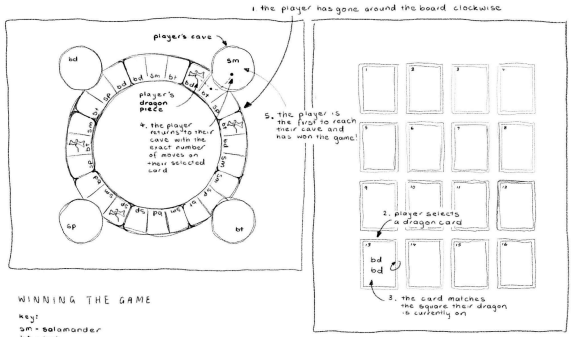


MOVING DRAGON TOKENS

key:
sm - salamander
bt - bat
sp - spider
bd - baby dragon

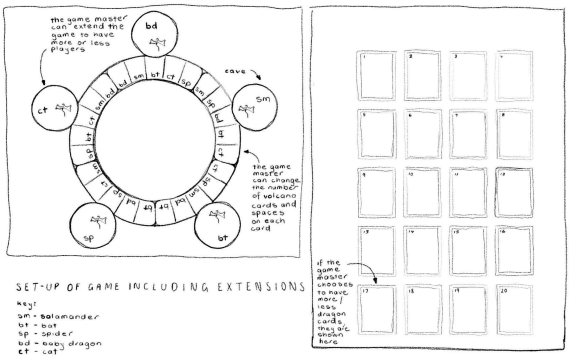
RETURNING TO THE CAVE

key:
sm - salamander
bt - bat
sp - spider
bd - baby dragon



WINNING THE GAME

key:
sm - salamander
bt - bat
sp - spider
bd - baby dragon



SET-UP OF GAME INCLUDING EXTENSIONS

key:
sm - salamander
bt - bat
sp - spider
bd - baby dragon
ct - cat

References

[1] GeeksforGeeks, “Introduction to Java Swing,” *GeeksforGeeks*, Feb. 15, 2022. https://www.geeksforgeeks.org/introduction-to-java-swing/

[2] “Instructions • 说明书 • Spielanleitung • Règle du jeu • Instrucciones 炽热巨龙 · Drachenstark Fort comme un dragon · Fuerza de dragón.” Accessed: Mar. 26, 2024. [Online]. Available: https://games4you.rs/sites/default/files/Fiery%20Dragons%20pravila%20igre.pdf