SPRINT 1 DOCUMENTATION

FIT3077 Architecture and Design

Team

CL_Monday06pm_Team069

Team Name

“There’s nothing to see here”

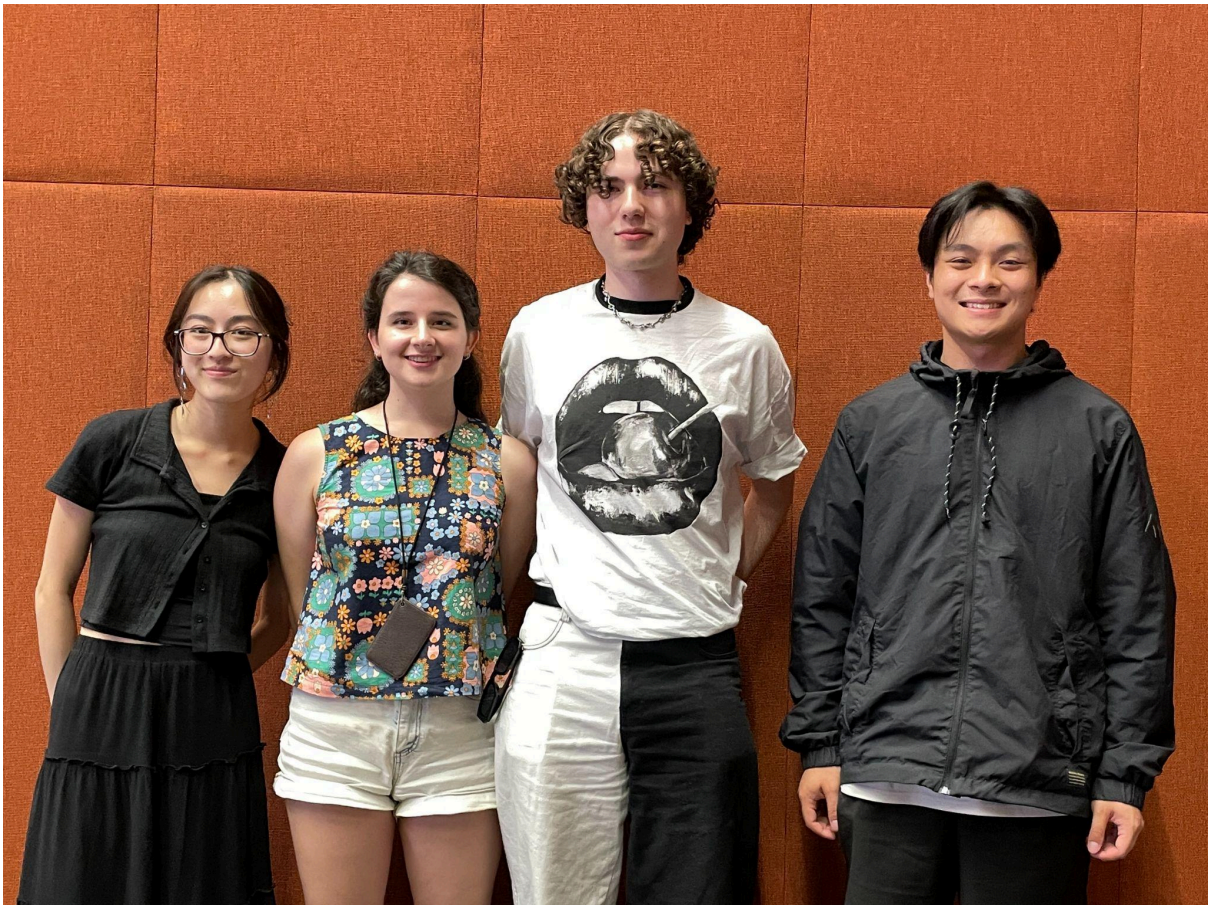
Team Members

Georgia Kanellis

Jun Hao Ng

Audrey Phommasone

Tye Samuels



Team Information

Team Membership

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 floating 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

- [x] ~~Document your team's regular meeting schedule and regular work schedule.4~~
- [x] ~~Document how the workload will be distributed and managed within your team.~~

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

- ☐ Document what programming language, APIs, and technologies are you planning to use and how this maps to the team's current expertise, and which ones you anticipate needing support from the teaching team.
- ☐ Justify your team's final choice of technologies that will be used.
- ☐ NOTE: we recommend that you do some very basic prototyping with your chosen technologies to ensure (i) everybody in the team knows how to use them (not just in theory, also in practice!) and that (ii) you can create an executable - something that will be needed for sprints 2 to 4. Ideally, you test that an executable you have created can actually be run on another computer (that does not have your set-up). You want to get this out of the way so that there will be no surprises for any of the following sprints.

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

- [ ] Submit a list of user stories (20 to 25 stories) that covers both the basic Fiery Dragons
  - [x] ~~Audrey - 5~~
  - [x] ~~Georgia - 7~~
  - [x] ~~Jun - 5~~
  - [ ] Tye - 5 (5 remaining)
- [x] ~~gameplay and initial ideas for your own extensions. A majority of the user stories are expected to be devoted to the basic requirements for the Basic prototype.~~

GAME FUNCTIONALITY / GENERAL

1. As a player, I want to be able to enter my age so that the starting player can be selected. (Georgia)

DRAGON CARDS

1. As a player, I want to be able to select a dragon card so that my character can progress in the game. (Audrey)
2. 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. (Audrey)
3. 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. (Jun)
4. 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. (Georgia)
5. 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. (Jun)
6. 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. (Jun)

BOARD / VOLCANO CARDS

7. 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 check how many spaces I have left until I am back into my cave so that I can check if I am able to move into the cave card. (Georgia)
8. 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. (Audrey)
9. 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. (Audrey)

-

10. As a player, I want to be able to check if a game space is already occupied so that I can tell if I am able to move to that space. (Georgia)
11. As a player, I want to be able to identify the species in the space I am currently occupying so that I know which cards to aim for. (Georgia)

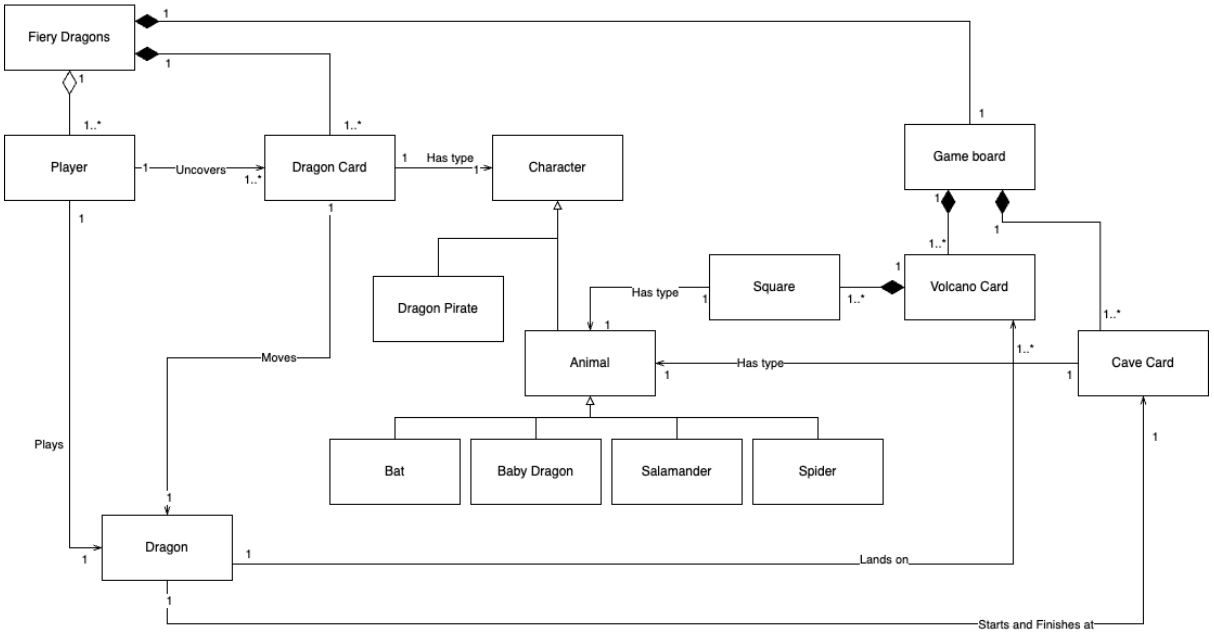
EXTENSIONS

2. 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. (Georgia)
3. 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. (Georgia)
4. 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. (Jun)
5. 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. (Audrey)
6. 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. (Jun)
7. 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. (Tye)

-

Domain Model

- [x] ~~Design and draw a domain model that covers both the basic Fiery Dragons gameplay and the listed extension requirements specified above.~~
- [ ] Provide detailed justifications for the domain model that you come up with, with a focus on the following aspects:
    - [ ] Provide a justification for each chosen domain entity and their relationships.
    - [ ] Were there any specific choices that you had to make while modelling the domain and WHY?
    - [ ] Explain any assumptions you have made, as well as any other part of your domain model that you feel warrants a justification as to WHY you have modelled it that way.
- [x] ~~Please note that a Domain Model is NOT a UML Class Diagram - it does not have any functionality nor any attributes.~~



Domain Model Justification

Following requirements from the game instructions...

Extensions are not included in the base game domain model, but is 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

the listed extension requirements specified above.
Provide detailed justifications for the domain model that you come up with, with a focus on the following aspects:
- Provide a justification for each chosen domain entity and their relationships.
- Were there any specific choices that you had to make while modelling the domain and *WHY*?
- Explain any assumptions you have made, as well as any other part of your domain model that you feel warrants a justification as to *WHY* you have modelled it that way.

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.

- Player
  - One player plays as a single dragon character. In each turn, each individual player will uncover one or more dragon cards
  -
  - One player also uncovers one to many dragon cards every 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.
- Dragon
  - One dragon starts and finishes within a single cave card. This is a one to one association as only one dragon will ever occupy a single cave.
- Dragon Card
  - The dragon card has a one to one association with the Character, as each dragon has a one character type.
- 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.
- Animal
  - Animal is another entity that specifies the character domain entity, it itself is a generalisation of each specific animal entity. These include Bat, Baby Dragon, Salamander and Spider. In the future, these animals could be 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
-

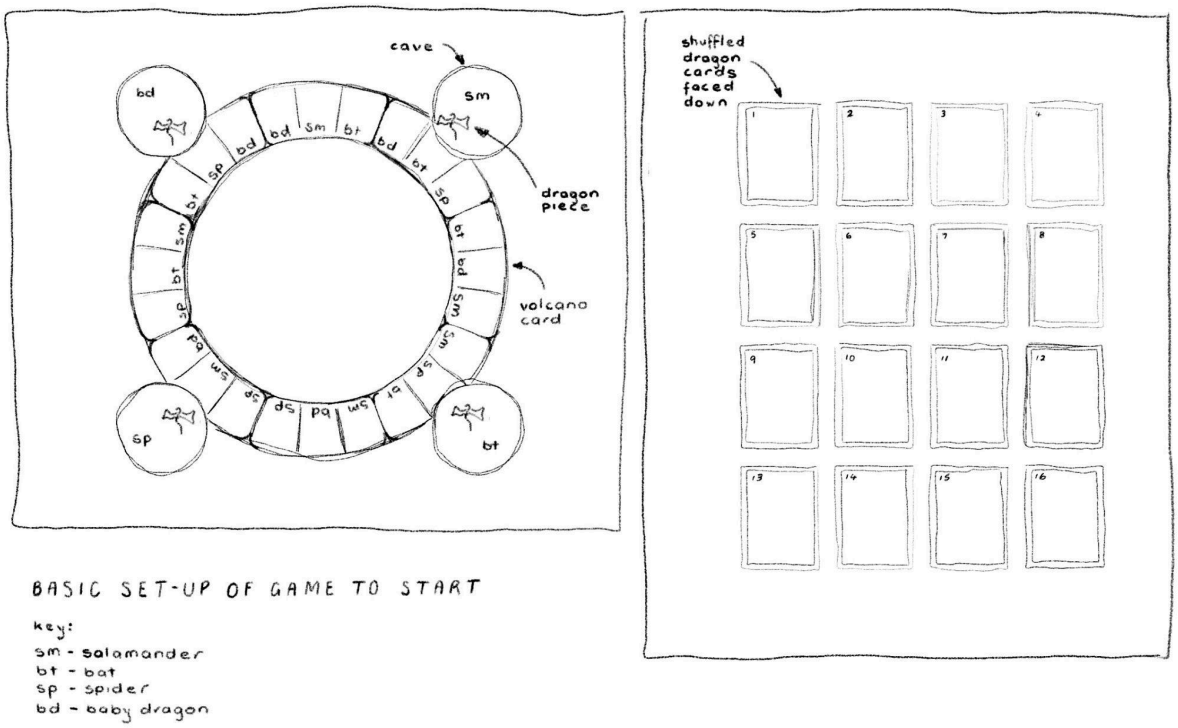
this entity to be one to many, even though the base game only would have 3 squares making up a volcano card. This decision was made to allow for the increase of squares within a volcano card to allow for game extensions.

- Game Board
    - As seen in the game rules, the game board is typically made up of eight volcano cards, and 4 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.
- Cave Card
    - One cave card has one animal type in each game, and thus has an association relationship between those two entities.

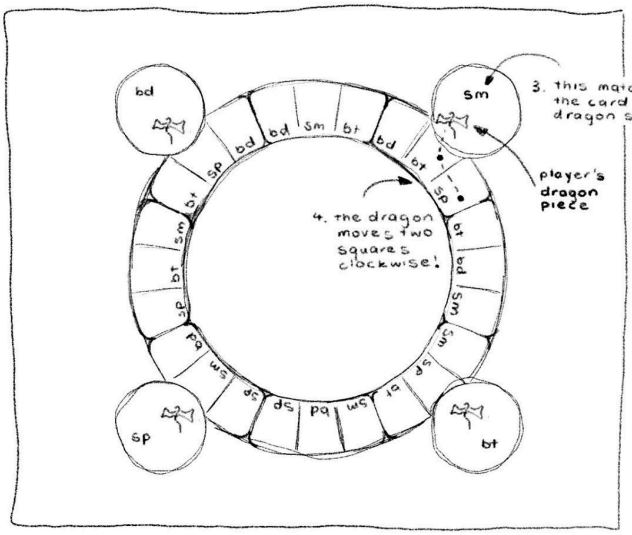
-

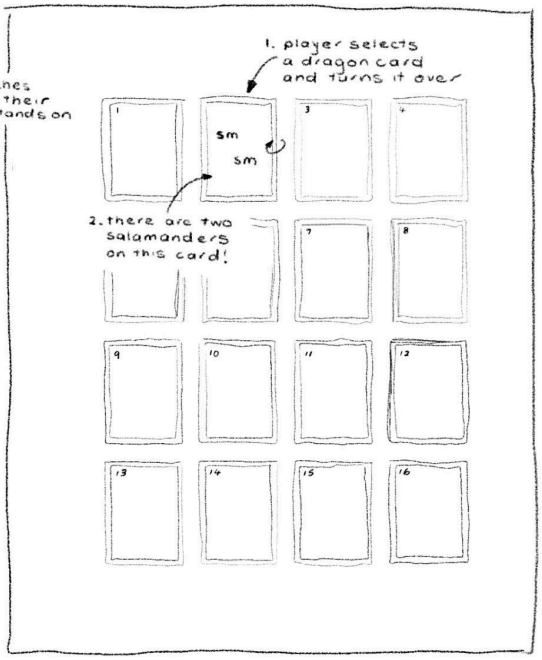
Basic UI Design

- [x] ~~Draw low-fidelity (low-fi) prototype drawings of the proposed user interface for the application.~~
- [x] ~~The low-fi prototypes need to demonstrate both the basic Fiery Dragons gameplay and the chosen extension requirements specified above. The prototypes should cover all the key interaction scenarios of the Fiery Dragon game (e.g., set-up of game to start, uncovering dragon cards of various types, moving of dragon tokens, winning situation) and the advanced feature(s) of your choice. This can be achieved in one large drawing space or across multiple pages. Avoid redundancy, that is, do not create multiple prototypes for the same interaction.~~
- [x] ~~All drawings should be large and clear enough to understand and any writing should be legible. You may use pen and paper, or digital drawing tools.~~



-

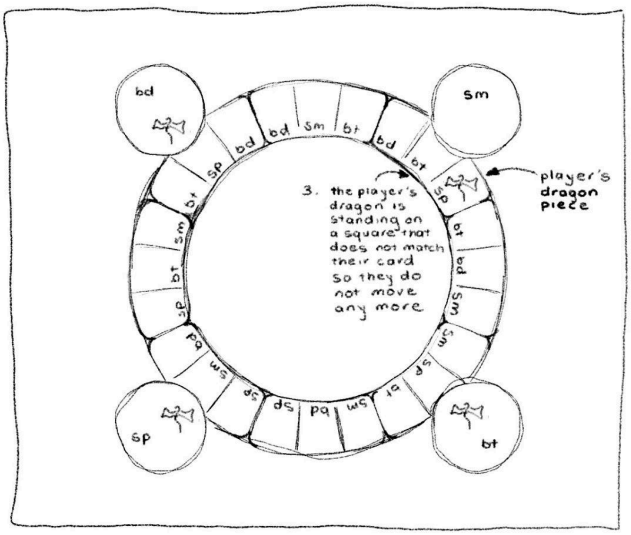


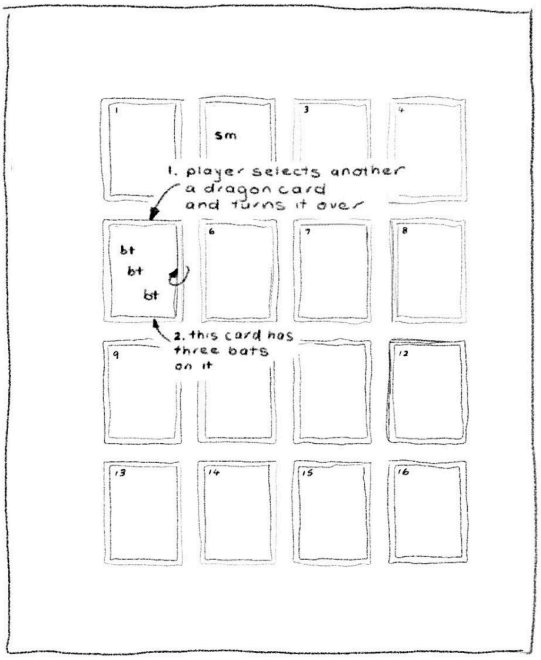


UNCOVERING DRAGON CARDS

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

-

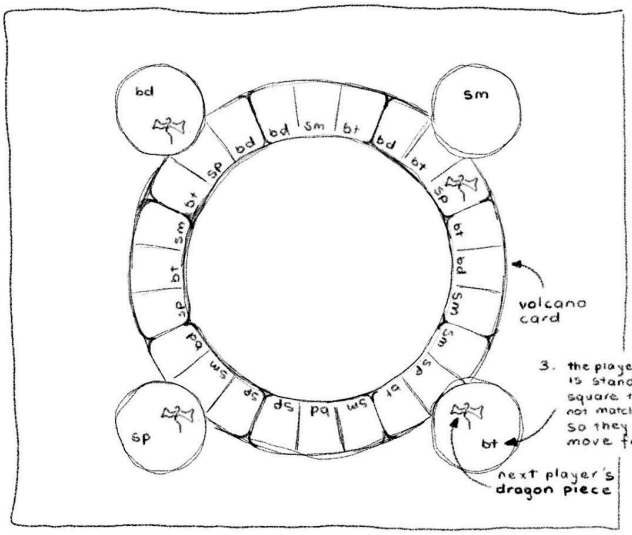


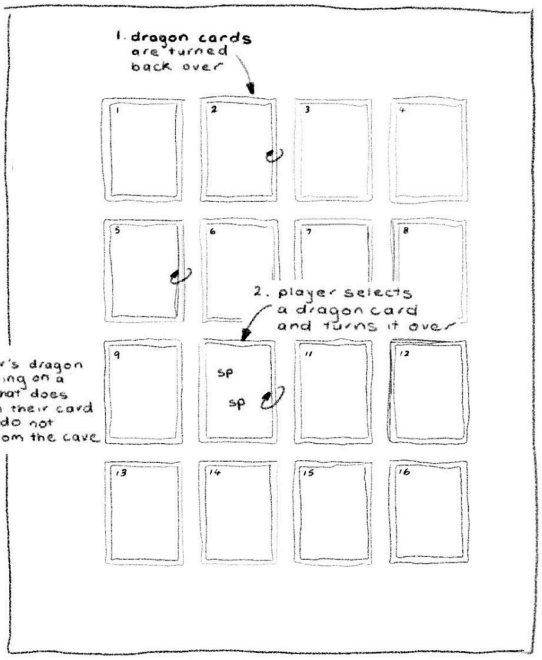


UNCOVERING DRAGON CARDS (PT. 2)

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

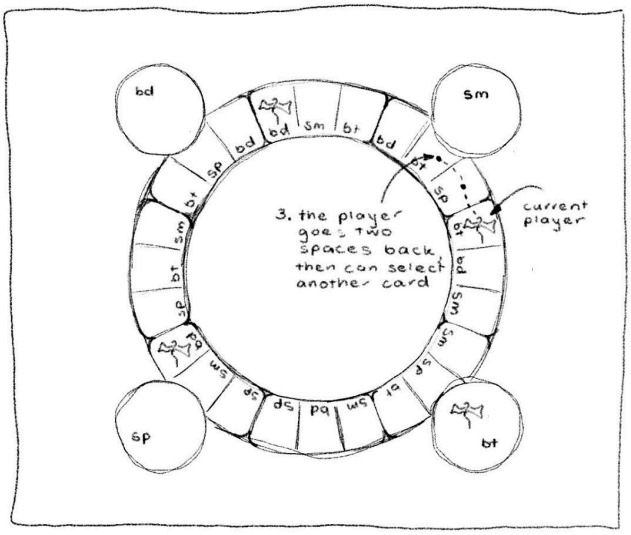
-

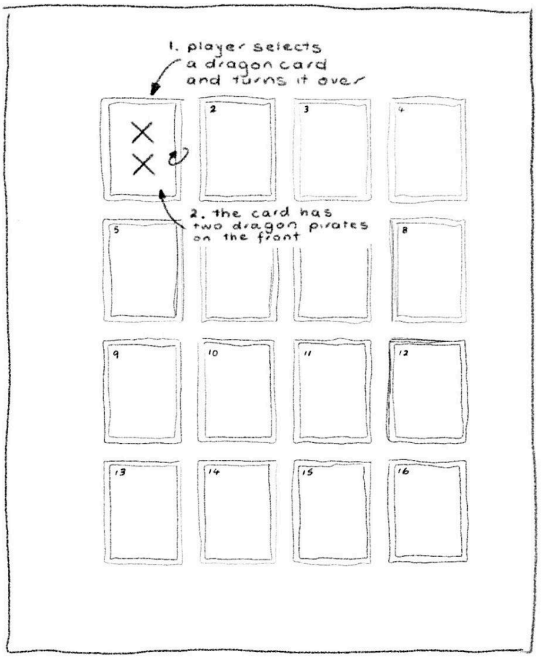




UNCOVERING DRAGON CARDS (PT 3)

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

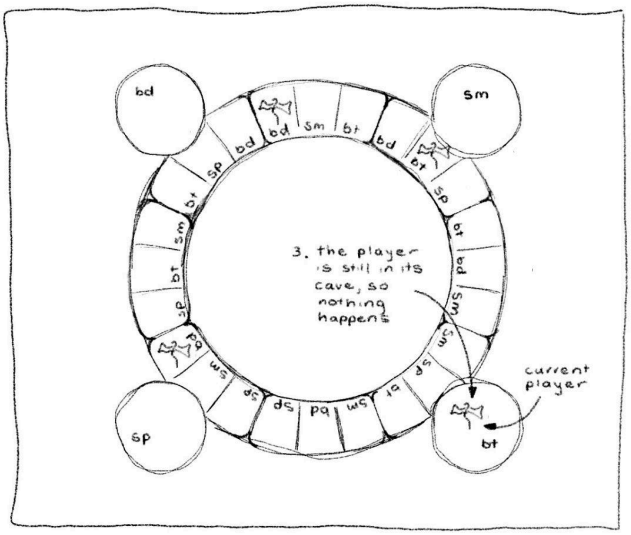


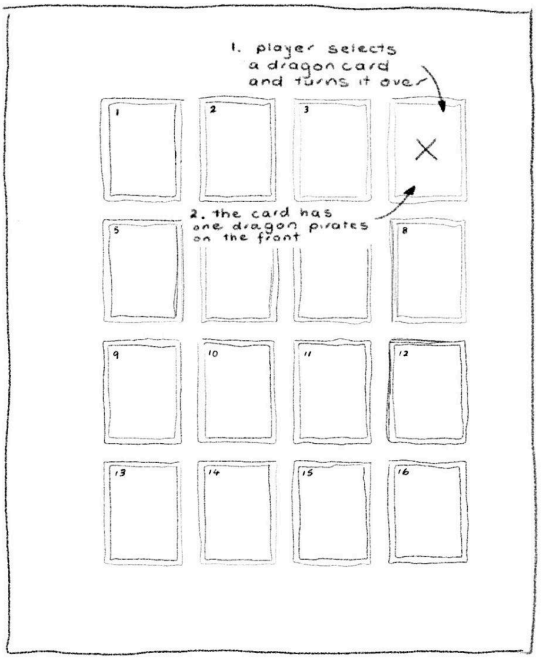


UNCOVERING PIRATE DRAGON CARDS

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

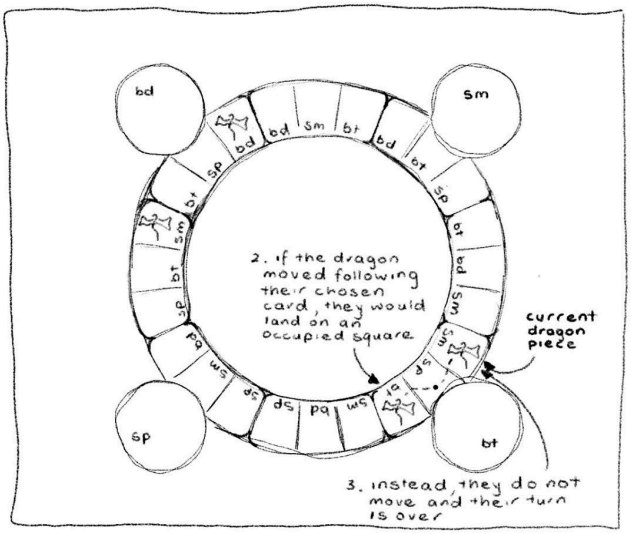
-

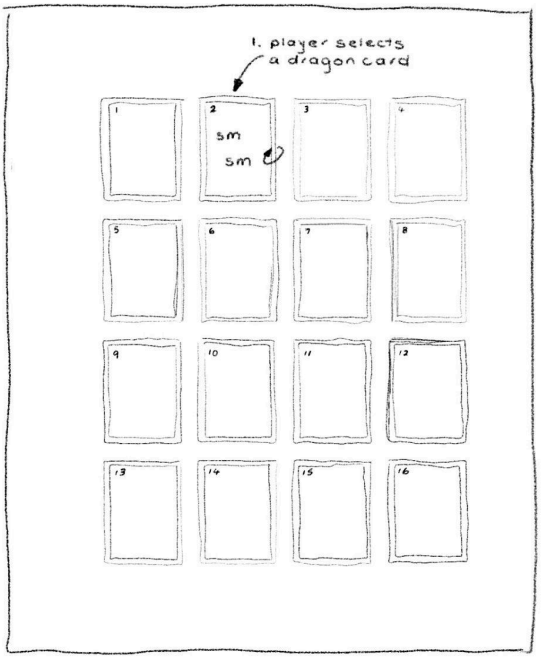




UNCOVERING PIRATE DRAGON CARDS (PT 2)

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

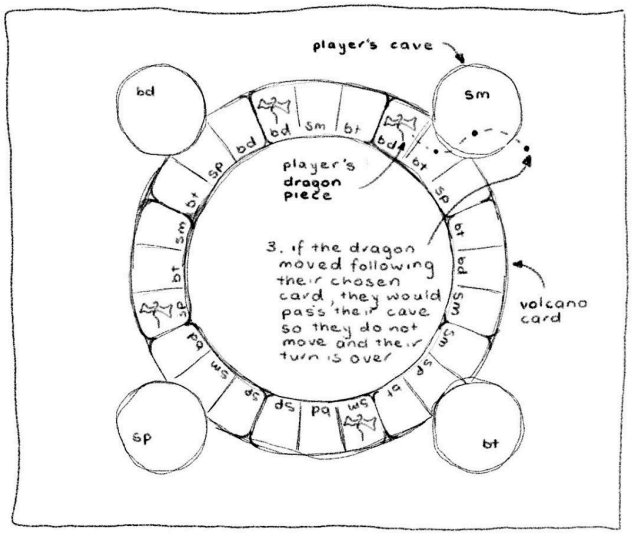


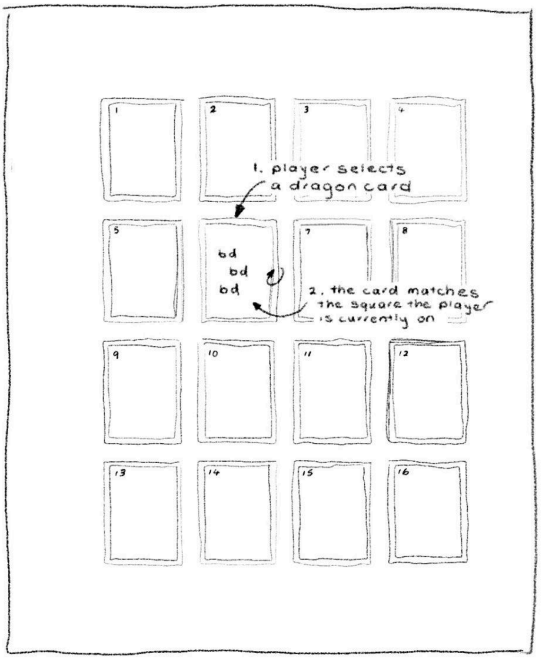


MOVING DRAGON TOKENS

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

-

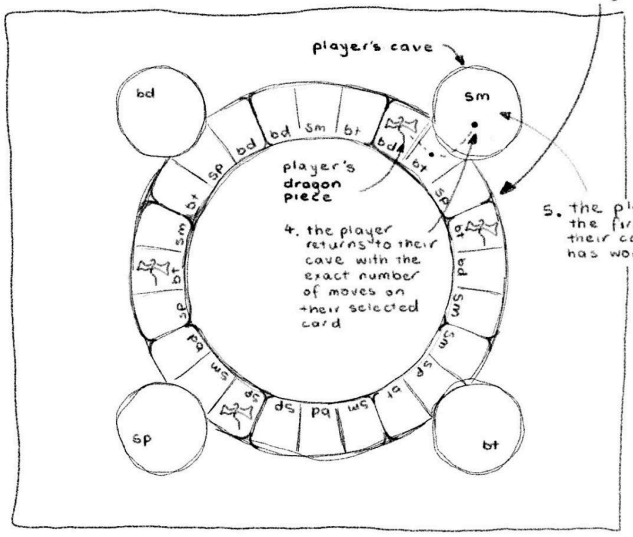




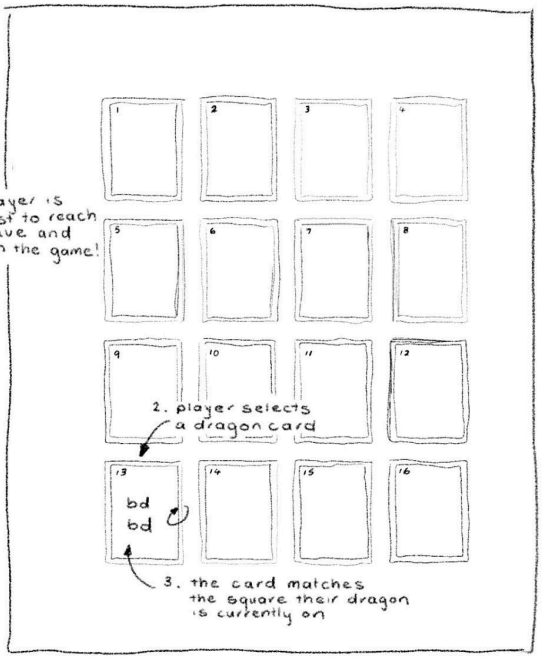
RETURNING TO THE CAVE

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

1. the player has gone around the board clockwise



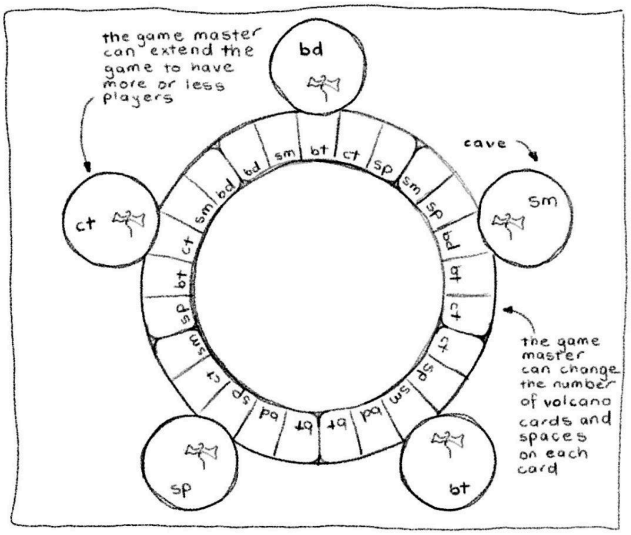
5. the player is the first to reach their cave and has won the game!

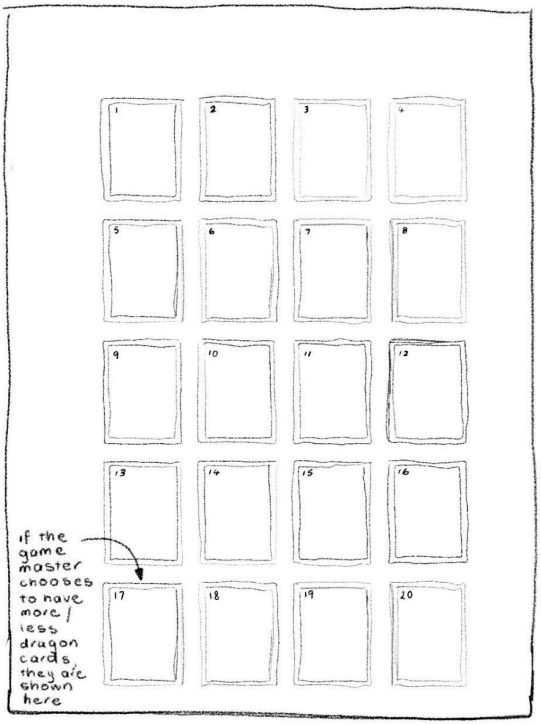


WINNING THE GAME

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/

-