Semester 1, 2024

FIT3077 – Software engineering: Architecture and design

Sprint 1: Team Information, User Stories, Domain Model, Basic UI Design

Team: Chitbusters

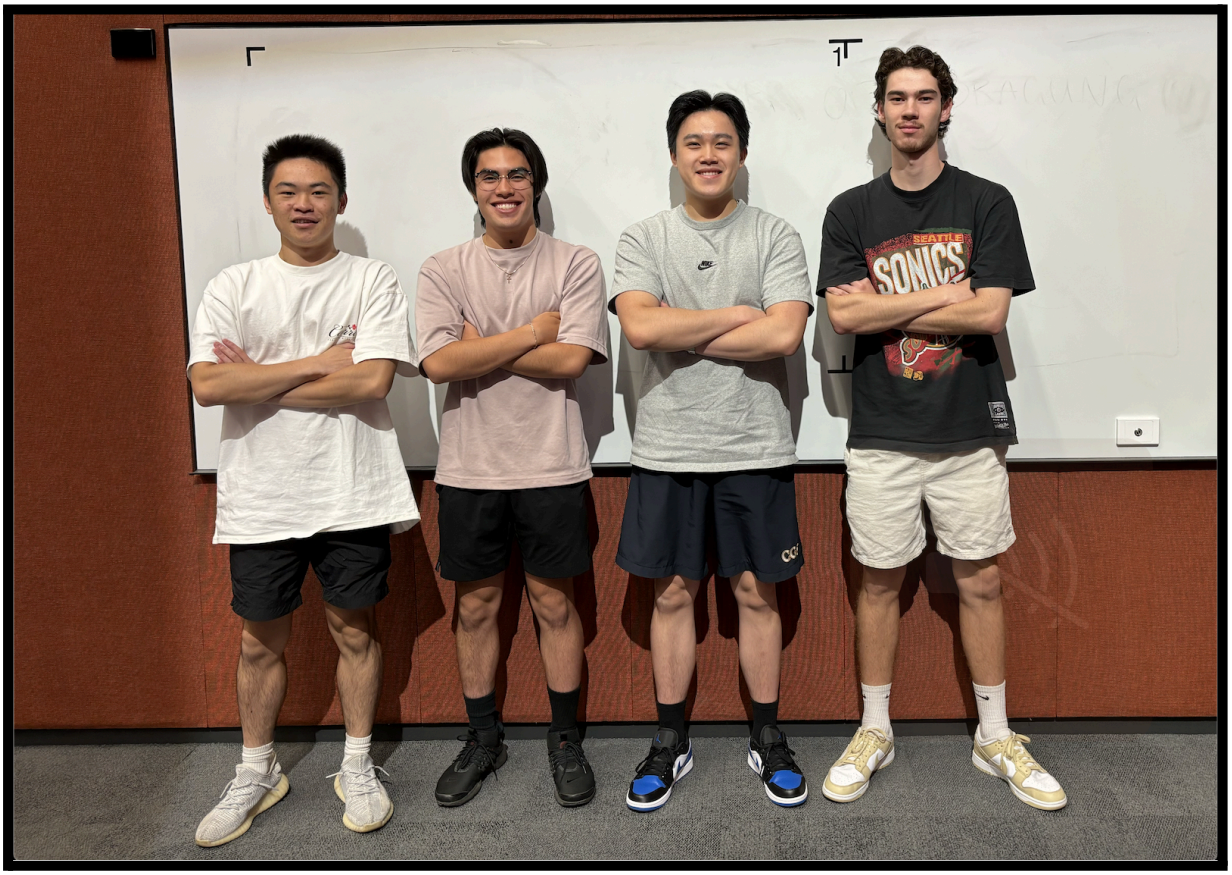
Date of Submission: 27th March 2024

Table of Contents

Team Information **3**
Team Members 3
Team Schedule 4
Technology Stack and Justification 5
Programming Language: Java 5
User Interface: Text-based 5
Discarded Alternatives: Python & Graphical UI 6
User Stories **7**
User Interface 7
Gameplay & Rules 7
Extensions 8
Domain Model **9**
Entities and Justification 10
Player 10
Cave 10
Dragon 10
Game Board 10
Chit Card 10
Volcano Card 11
Dragon Skull 11
Animal 11
Tile 11
Basic UI Design **12**
Starting Menu 12
Starting Game Board 13
Successful Match 14
Not Matching 15
Player Collision 16
Choosing Dragon Skull 17
Almost Wins 18
Player Wins 19
Shuffling Dragon Cards 20

Team Information

Team Name: Chitbusters



Team Members

Brandon Luu
Student number: 32500793
Email: bluu0013@student.monash.edu
Strengths: Java, Domain Modelling
Fun Fact: Never broken a bone before

Max Zhuang
Student number: 33109435
Email: mzhu0033@student.monash.edu
Strengths: Testing, Logic, Java, Working in Agile
Fun Fact: Owns a golden retriever

Jeremy Ockerby
Student number: 33157375
Email: jock0003@student.monash.edu
Strengths: Algorithms, OOD, version control tools
Fun Fact: Fractured my elbow in a hurdles race

Raymond Li
Student number: 33119937
Email: rlii0089@student.monash.edu
Strengths: Python, Java, UI Design, Debugging
Fun Fact: Have fractured, but not broken a bone

Team Schedule

Prior to initiating the project, our team utilised the time given in the workshops to outline weekly objectives and address queries with teaching assistants. We established and scheduled a recurring meeting which was held during the week as per all members' availability. This would allow everyone to participate and contribute equally and fairly. Documented below is our team's contribution log which provides the dates when meetings were held, as well as the completion dates for each task. Through the log, we aim to offer transparency into our team's progress.

Link to Google sheets document (Contribution Log): FIT3077 - Contribution Log

Work will be distributed at the beginning of each sprint, where a "sprint planning" meeting will be organised with all team members in attendance to discuss the specific tasks that need to be completed in the current sprint. After analysing what needs to be done, the workload will be evenly distributed with team members having the option to pick out which specific tasks they would like to complete according to their strengths and weaknesses.

Technology Stack and Justification

Programming Language: Java

Justification: When identifying a suitable programming language, we decided 3 criteria had to be met:

- **Familiarity:**
  Based on discussions with the team, it was unanimously concluded that all members' current expertise includes familiarity with Java. Choosing Java as the programming language will align with the team's existing skills and knowledge. The familiarity can accelerate the development process and prevent the learning curve that is associated with adopting a new programming language.

- **Object oriented programming:**
  Java is known for supporting object oriented programming principles. The brief emphasises object oriented approaches and practices, and Java provides this by allowing the implementation of classes, objects, inheritance, polymorphism and encapsulation. These OOP features will be used to model parts of the board game such as players, game board, chit cards and game logic. Especially for future sprints where extensions of our existing program need to be made, Java's design principles allow for scalability and maintainability.

- **Platform Compatibility:**
  Since the final product requires the game to be run on any machine without extra libraries, Java is an ideal choice for cross platform development. Some of the members use macOS whilst others use Windows. The game can be deployed on multiple platforms without modification which aligns with the objective of creating a game for a wide audience.

User Interface: Text-based

Justification: The decision to maintain a text-based user interface for our game is based on a unanimous decision after evaluating the project objectives and constraints. While graphical interfaces can enhance user experiences, opting for a text-based interface offered several advantages for our project:

- **Prioritise gameplay:**
  It allows us to prioritise on gameplay to ensure a robust experience within the constraints of our project timeline.

- **Compatibility:**
  By using a text-based interface, it ensures that the game is compatible on all systems as it eliminates the need to rely on external resources such as images, which may not always be available or compatible.

- **Nostalgia and immersion:**
  By using a text-based interface, we aim to evoke nostalgia through a simple, yet classic text-based game.

Discarded Alternatives: Python & Graphical UI

We considered Python, however we decided against it for various reasons:

- **Performance:**
  Java is typically faster than Python especially for complex computational tasks that game development might require.

- **Static typing:**
  Java is designed to be statically typed so that variables must be declared with their data types. This can help with catching errors at compile-time rather than run-time. By doing so, this can lead to more robust and reliable code which is important to not negatively affect the user experience.

- **Resources:**
  Java has a large community and extensive documentation and resources available. This can be helpful for troubleshooting issues or seeking guidance from teaching assistants and experienced developers.

Having a graphical User Interface was considered for the game. However, the team opted out of this approach for a few reasons:

- **Familiarity:**
  Most members are not familiar with creating a game engine connected to a graphical interface. Learning this may take an unnecessary amount of time, depending on the person, which can potentially cause time crunch issues later down the line.

- **Creating/Sourcing Graphics:**
  The graphics themselves can serve to be an issue given that creating them ourselves may not be feasible for the given time frame. Alternatively, we could find sources for the graphics, however including them into the implementation can serve to add additional unwanted issues if they do not work. Adding extra content to the implementation can create extra pathways for debugging, which can increase the time taken to produce the game.

User Stories

User Interface

These user stories mainly focus on the user experience. They aim to outline the requirements for the interface of the game, including aspects such as game notifications, how to distinguish between characters, ease of access, etc.

1. As a player, I want to know when my turn is, so that I can select a dragon card to move my character.
2. As a player, I want to be able to see the location of other players' characters, so that I know how each player is progressing.
3. As a player, I want to know where my dragon is on the board, so that I know which dragon card I need to select in order to move.
4. As a player, I want to be able to choose from a selection of chit cards, so that I can progress along the board.
5. As a player, I want to be able to see all previously selected cards in my turn, so that I know I cannot pick those cards again this turn.
6. As a developer, I want to know the ages of all players, so that the playing order can be determined.
7. As a player, I want to be able to easily distinguish between the gameboard, the animals and dragons, so that I can focus on gameplay.
8. As a developer, I want all selected cards to be hidden after a player's turns, so that players are not at a disadvantage by revealing cards to other players.
9. As a player, I want to be able to read the rules and/or instructions at any time, so that I can be aware of what I can and cannot do.

Gameplay & Rules

These user stories mainly focus on how the game will begin, as well as how it should play out. They will also involve those relating to the rules of the game, how they are enforced, and the result of the game ending. They will touch on rules that allow you to move again and why you weren't able to move.

10. As a player, I want everyone to receive a victory message after I return to my cave, so that all players know the game has ended.
11. As a developer, I want a player's character to move only if a matching animal card with the current position is picked, so that the game operates in compliance with the game rules.
12. As a developer, I want there to be a punishment when a dragon skull card is picked, so that there is a risk for poor memory.
13. As a player, I want to have a clear indication of why I cannot move, so that I won't be confused as to why my turn has ended.

14. As a player, I want to land on the animal with a matching card I know, so that I can move again.
15. As a developer, I want to ensure all caves are evenly spaced around the board, so that all players have the same distance between them at the beginning of the game.
16. As a player, I want to be able to see the locations of all the caves, so that I know my starting point as well as my destination

Extensions

The user stories in this section are based around some potential extension ideas that could be implemented to 'shake things up'. These changes should change the way the game is played, and add in extra challenges for players to overcome throughout their playthrough.

Potential extensions, and their relevant user stories, are as follows:

I. If the same card is chosen three times, all the dragon cards are shuffled

17. As a player, I want the game to shuffle the cards when the same card is chosen 3 times, so that I can be challenged to remember the new locations of the cards.

II. Change the board depending on the number of players (change number of volcano cards accordingly)

18. As a developer, I want to know how many players are participating, so that the board can be generated with a varying number of volcano cards based on the number of players.

III. Extending the game board

19. As a player, I want the game to have a choice of extending the game board by increasing the number of volcano cards, so I can choose how long I want the game to be played for.

IV. Add additional animals

20. As a player, I want the game to have more animals, so that I can increase the difficulty of the game by having more chit cards to choose from.

V. Add dragon hunter card that sends player back to their cave

21. As a player, I want to try to avoid the dragon hunter card, so I don't get sent back to my cave.

VI. Players that land on the same tile will have to 'fight' for who gets to remain on the tile. Players will take turns revealing cards, and the first player to reveal a dragon skull moves back the number of spaces indicated on the card.

22. As a player, I want to try to avoid picking the dragon skull card when landing on a tile with another player on it so that I don't move back the number of tiles on the dragon skull card.

Domain Model

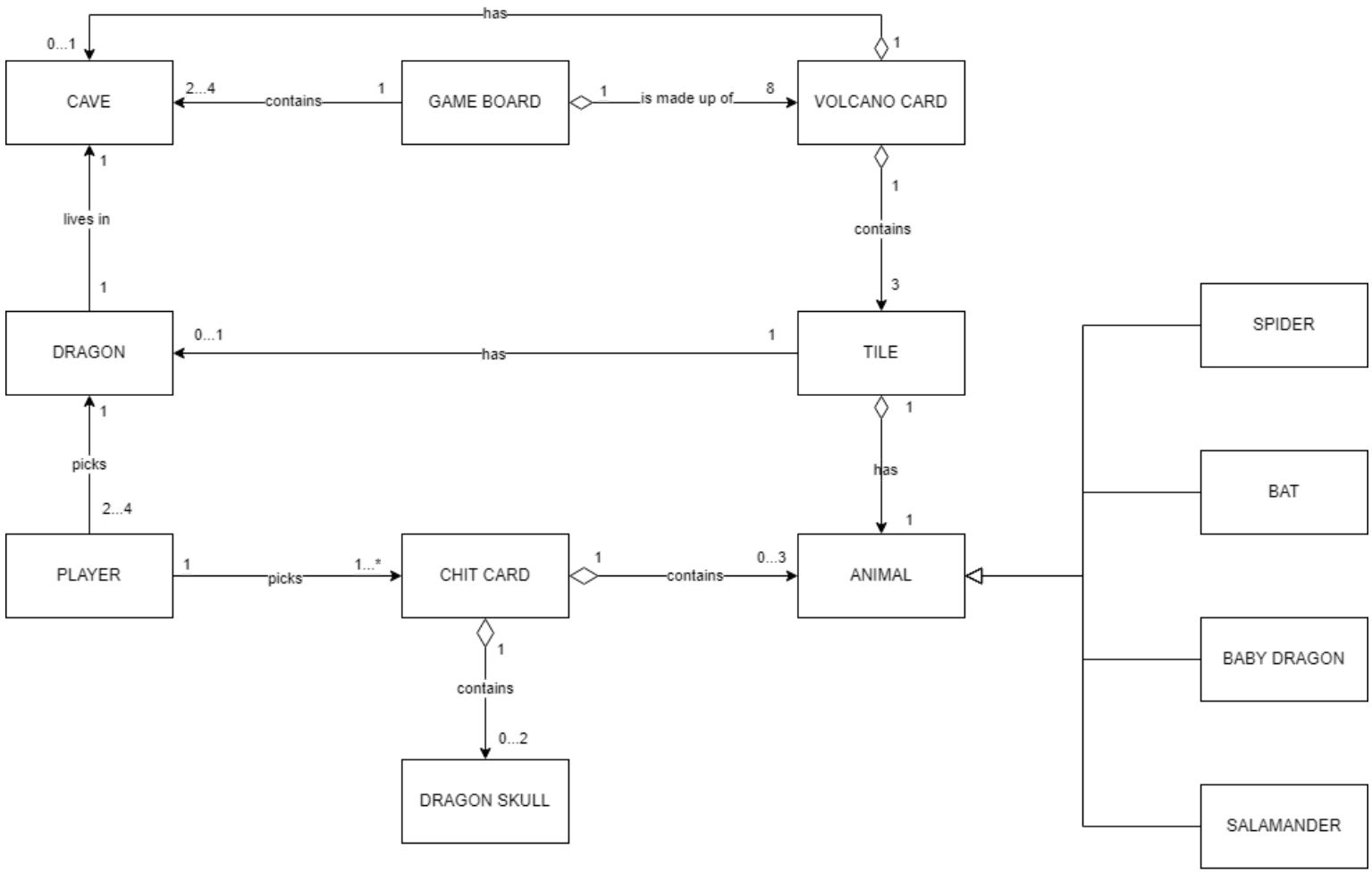


Figure 1. Domain Model of Fiery Dragons

Entities and Justification

Player

The player entity will be the user that plays the game. There needs to be a minimum of 2 players as the game rules require at least 2 to be played. There can be a maximum of 4, as stated in the game rules. Each player will pick one dragon character to play as at the beginning of the game, hence there is a many to one association between player and dragon.

Additionally, during the player's turn they can turn over a chit card, and the animal on the chit card corresponds to the animal the player's character is currently standing on, they can continue their turn and turn over another chit card, making it possible for a player to turn over multiple cards in one turn. Thus, there will be a one to many association between the player and chit card entity.

Cave

The Cave entity is the cave position in the game. It will have at least 2 as there will be one for each player, and the minimum number of players is 2. The maximum number of caves is also tied with the maximum number of players, which is 4.

Dragon

The dragon entity is a playable character picked by the players. Each player must pick a dragon and they can only pick one. This means that the number of dragon entities is tied with the number of players. The dragon will begin in its designated cave and during the game, will be located on a tile throughout the game, creating associations to the cave and tile entity.

Game Board

The Game Board is the playing field for Fiery Dragons. There will only be one of this entity per game/session. This entity will be made of exactly 8 volcano cards, and can contain 2 to 4 caves in one instance of the game. Thus the game board entity will have aggregations to the cave and volcano card entities.

Chit Card

The Chit Cards are the means which allow the players to move around the board. There are 16 available to be chosen, which can either contain Animals, having 1 to 3 of the same animal per card, or Dragon Skulls, having 1 or 2 per card. Players are able to pick up more than one Chit Card per turn, given they were allowed to move forward. As a chit card can contain either animals or dragon skulls, it will have an aggregation to the animal entity and an aggregation to the dragon skull entity.

- **Discarded Alternative:** An alternative implementation that was considered was using a generalisation relationship to animal and dragon skull instead of an aggregation relationship. However, as an animal itself is not solely a type of chit card, this idea was not put forward. Extending upon this, another alternative was to introduce two new entities: advancing cards and regressing cards, to act as the two types of chit cards with a generalisation to chit card and an aggregation to animal and dragon skull respectively. This was not pursued further to maintain simplicity and no clear additional benefits of introducing these two new entities for the basic rules of the game were found.

Volcano Card

Volcano entity is the tiles which make up the Game Board. There are exactly 8 Volcano cards in total with each card consisting of a combination of 3 tiles. Thus the volcano card entity will have an aggregation relationship to the tile entity. Additionally, a volcano card may also contain a cave creating another aggregation to the cave class, however not all volcano cards will contain a cave so the cardinality on the cave side will be 0 or 1.

- **Discarded Alternative:** An alternative implementation was connecting the volcano card entity directly to animal and not including the tile entity in the domain model. However, after further consideration we believe that an intermediary entity, being tile, is needed to help locate the position of the dragons.

Dragon Skull

Dragon Skulls, when revealed, make the player move backwards. This entity can only be found on Chit Cards, and can have either 1 or 2 Dragon Skulls per card.

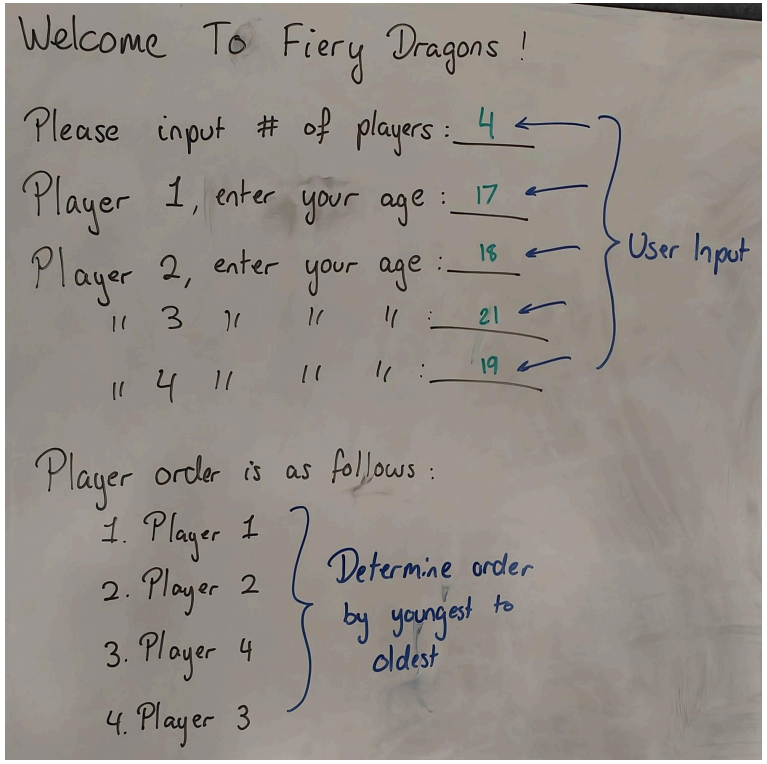
Animal

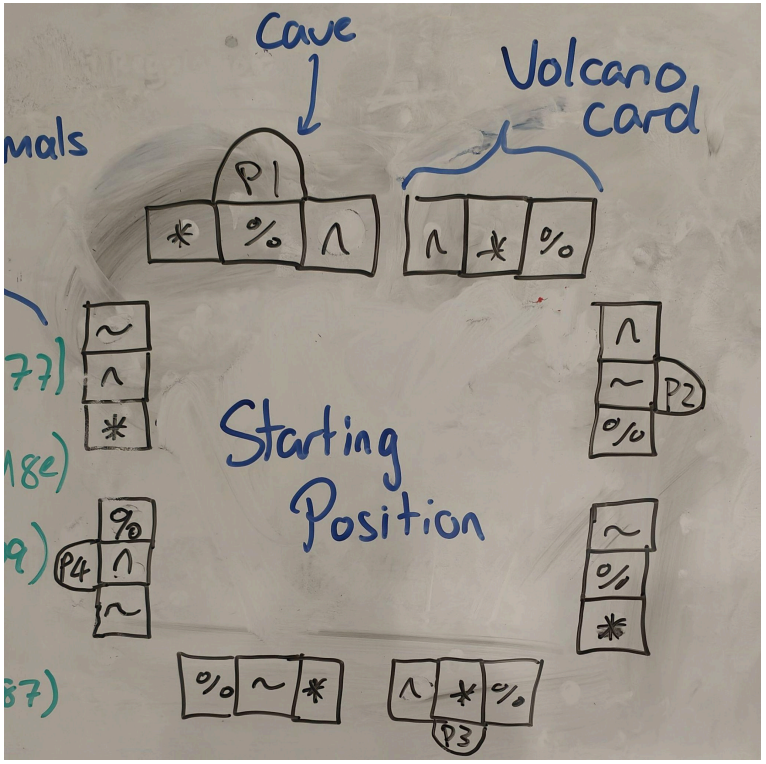
The Animal entity describes the 4 animals seen on the Volcano Cards, and Chit Cards. These animals are the Baby Dragon, Bat, Salamander and Spider, which all act as its own entity and have a generalisation relationship under the animal entity.

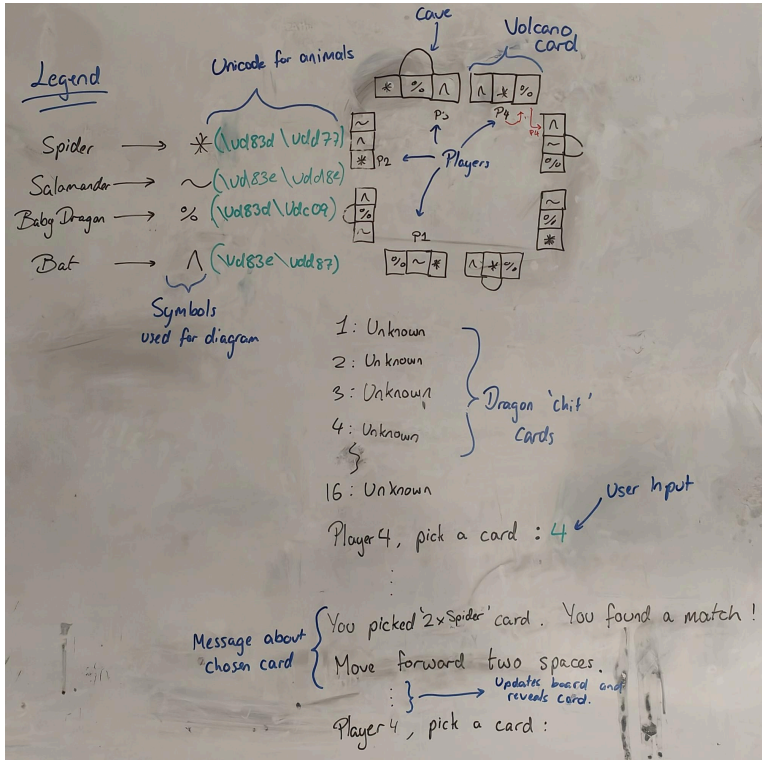
Tile

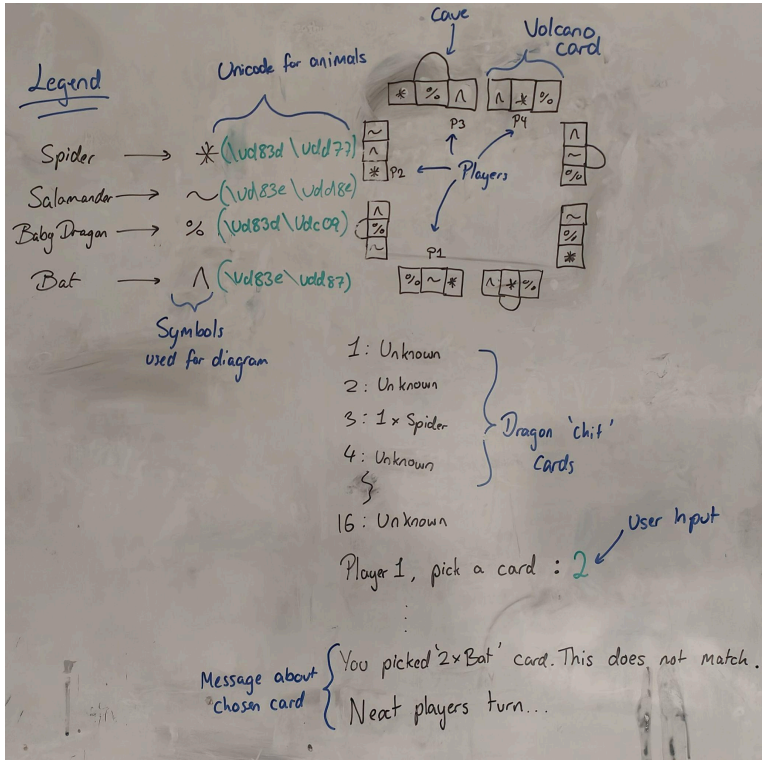
The tile entity is the specific position on a volcano card. Under the basic game rules each volcano card will consist of three tiles, and each tile will contain one animal. Thus, there will exist an aggregation relationship between the tile and animal entities.

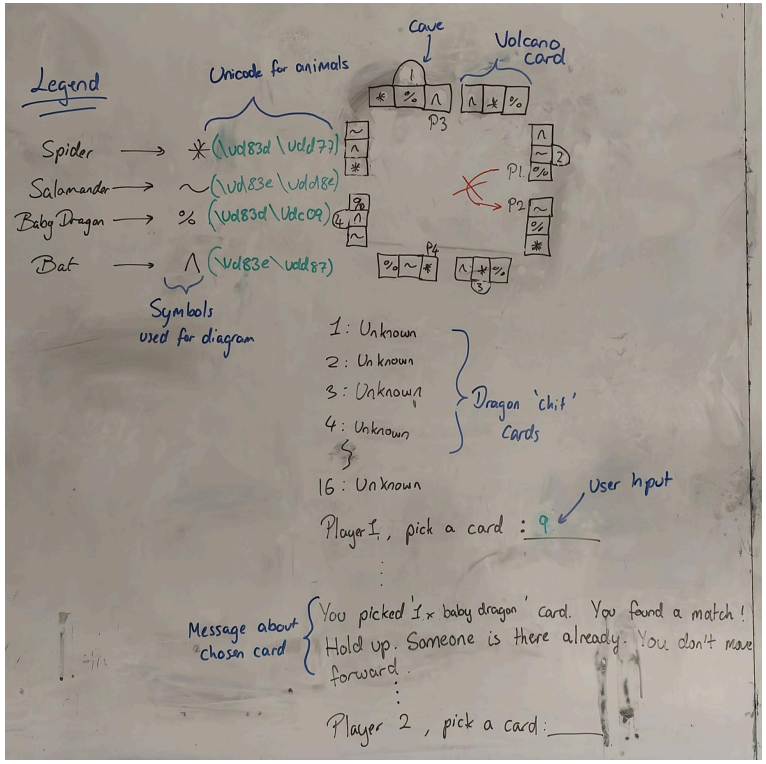
Basic UI Design

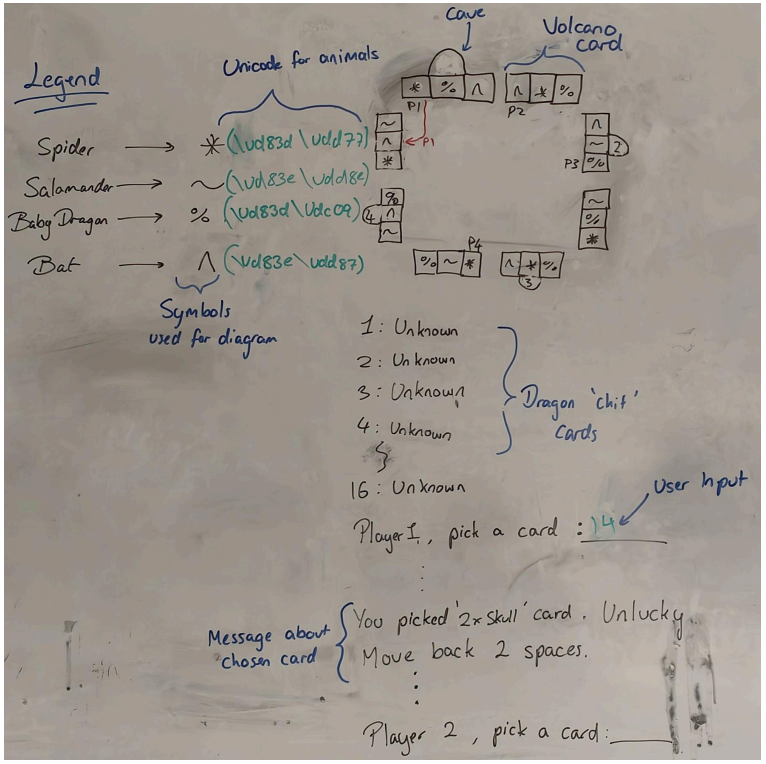
Title	UI Diagram	Explanation and Justification
Starting Menu	 **Note: Black indicates prints to console, blue is annotations, green (except for unicode) is user input, and red is player movement/s.*	In the starting menu, players are welcomed by a message printed through the console. They are then prompted to input the number of players that will be participating in the game. Once the number of players has been confirmed, each player will then be prompted to enter their age. This will determine the order of play, with the youngest player going first.

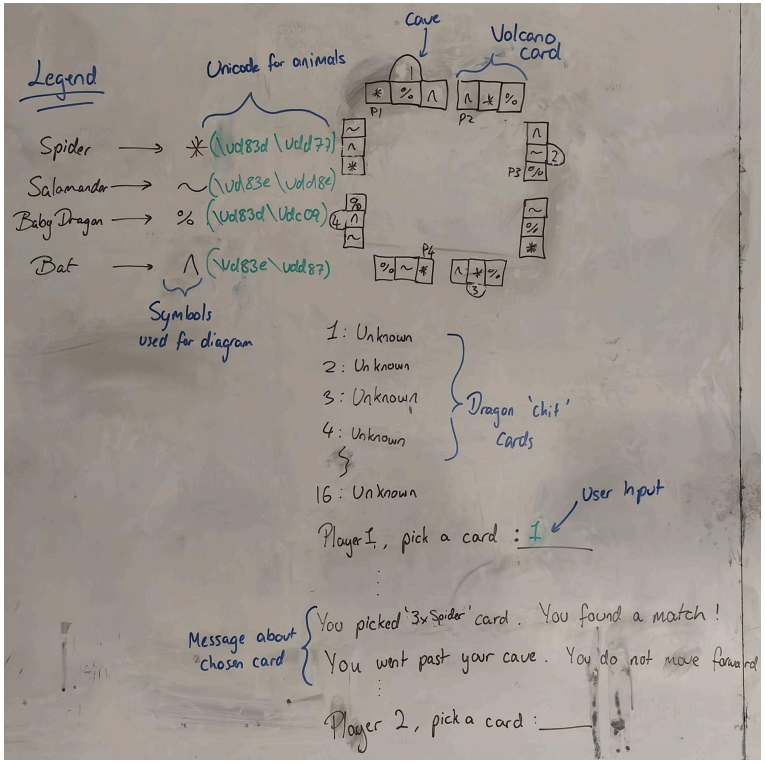
Starting Game Board	 **Note: Black indicates prints to console, blue is annotations, green (except for unicode) is user input, and red is player movement/s.*	Once the player order has been sorted out, the game board will then be generated and printed through the console. Players will start in their caves, labelled 'P1' for player 1, 'P2' for player 2, etc. Once they leave the cave, the caves will have the number corresponding to the player that was previously there. The players will then move around the board as 'P1', 'P2', etc.

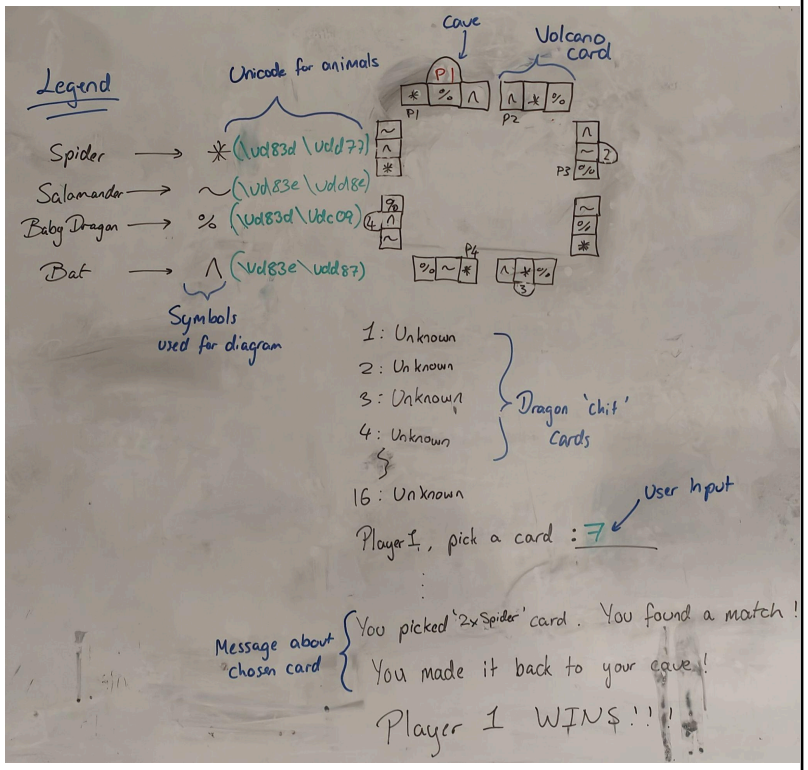
Successful Match	 **Note: Black indicates prints to console, blue is annotations, green (except for unicode) is user input, and red is player movement/s.*	For all following diagrams, they will look very similar as these are examples of what happens during the game. Each turn will look very similar to this where the game board and dragon cards will be printed to the players. It will then ask for the player to choose a card. In this diagram it shows what happens when a player picks the correctly matching card. Below the user input section, a printed message will reveal the card that was chosen and that it was a match to the player's current tile. After that, indicated by the three dots, it will update the board and allow the player to go again.

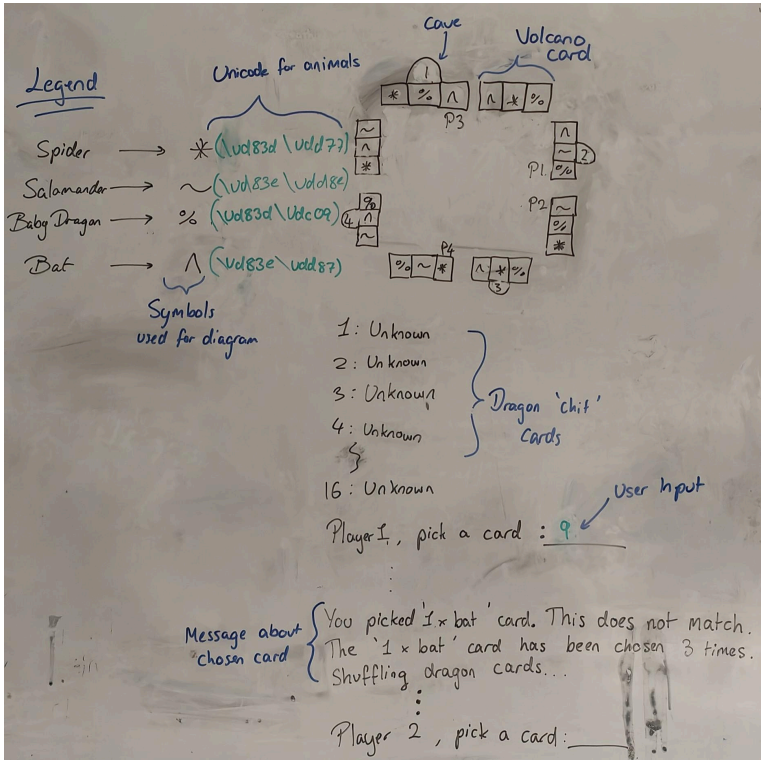
Not Matching	 **Note: Black indicates prints to console, blue is annotations, green (except for unicode) is user input, and red is player movement/s.*	In this diagram, it shows the outcome of an incorrect match. The console will once again reveal the card, but will also announce that it is not a match. It will then announce that it is the next player's turn, after which it will refresh the board. The messages regarding what card the player has picked and the content of the card are deleted from the console in preparation for the next player's turn so that players can not look back at previous turns to see the contents on the chit cards. Additionally, this diagram shows an example of a player that has already moved in the same turn. This is evident by the dragon card labelled '3: 1 x Spider' as this will only appear when the player has successfully matched and moved in this current turn. All revealed cards will then be hidden after this player's turn has finished.

Player Collision	 **Note: Black indicates prints to console, blue is annotations, green (except for unicode) is user input, and red is player movement/s.*	In the event of a player successfully matching the dragon card with their current tile, but cannot move due to another player being on the tile, the console will print that you have matched the card, but will also announce that you cannot move since there is someone currently there. It will then move to the next player's turn after printing the board and dragon cards again.

Choosing Dragon Skull	 **Note: Black indicates prints to console, blue is annotations, green (except for unicode) is user input, and red is player movement/s.*	If a player has chosen a dragon skull card, the console will announce it, then indicate how many spaces the player will move back. The board will be updated, chit cards will be hidden, and the next player's turn will start.

Almost Wins	 **Note: Black indicates prints to console, blue is annotations, green (except for unicode) is user input, and red is player movement/s.*	Similar to 'Player Collision', players will not move forward if the card selected would cause the player to go past their cave. Here, 'P1' needs '2 x Spider' to make it into the cave. Since they picked a '3 x Spider', the console will announce that they match the card, however cannot move forward since they would go past their cave.

Player Wins	 **Note: Black indicates prints to console, blue is annotations, green (except for unicode) is user input, and red is player movement/s.*	This diagram demonstrates a winning scenario where Player 1 is two tiles away from reaching their cave. The user picks the card in the 7th position by entering 7 as an input and finds a chitcard with two spiders. This matches the spider tile that Player 1 is currently on and means they can move forward by two tiles. A player can only win if they move by the exact number of tiles that are required to reach the cave. After picking position 7, the player is shown a message reflecting the pick and announcing the winner of the game.

Shuffling Dragon Cards	 **Note: Black indicates prints to console, blue is annotations, green (except for unicode) is user input, and red is player movement/s.*	This diagram shows an example of what could happen for one of our proposed extensions, which shuffles the position of all chit cards if one card has been picked three times. This will also cause all currently revealed chit cards to become hidden again, even if it is in the same player's turn. When this happens, the console will announce which card has been picked three times, then shuffle the cards.