FIT3077: Software Engineering: Architecture & Design

Sprint One Deliverables

Christine Chiong, 31985270
Jensen Kau, 33053332
Wong Yi Zhen Nicholas, 32577869

Table of Contents

1. Team Information

Team Name: CoolBeans

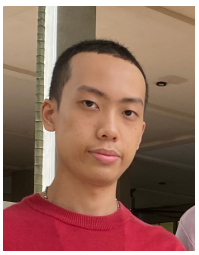
Team Photo

From left to right: Wong Yi Zhen Nicholas, Jensen Kau, Christine Chiong

Team Membership

Name: Christine Chiong
Email: cchi0067@student.monash.edu
Contact Number: +60109339667, **Discord:** kewisseh#8482
Technical Skill:
Christine is proficient in the languages Python, Java, JavaScript, TypeScript, HTML, CSS, SASS, and SQL. Christine also has experience in the Express framework as well as React and Material UI libraries. For UI/UX Design, Christine has worked with design platforms including Canva and Figma.
Professional Strength:
Christine is a team-oriented individual and enjoys working with highly-motivated teams to deliver goals together. Christine is also ambitious and strives for the best in her work.
Fun Fact:
Christine likes to draw in her free time and has experience working with digital art. Christine also enjoys watching anime and Genshin Impact in her free time.

Name: Jensen Kau
Email: jkau0039@student.monash.edu
Contact Number: +60172286088, **Discord:** Idle#0142
Technical Skill:
Jensen is proficient in the languages Python, Java, and C#. Jensen also has experience in the Django framework, the Unity game engine, and dockerizing applications.
Professional Strength:
Jensen is a very quick learner, being able to grasp new concepts which contributes to his flexibility in performing tasks and strong problem-solving skills. Jensen is also a communicative individual with no problem in expressing his thoughts or responding in a short amount of time.
Fun Fact:
Jensen is the youngest in our team which means that he has a lot of potential to grow. Jensen also enjoys watching anime and playing video games in his free time.



Name: Wong Yi Zhen Nicholas
Email: ywon0083@student.monash.edu
Contact Number: +601139391351, **Discord**: bon bon#0966
Technical Strength:
Nicholas is proficient in the languages Python, Java, JavaScript, TypeScript, HTML, CSS, SASS, Perl, and SQL. Nicholas also has experience in frameworks TailwindCSS, Bootstrap, Angular, Express, and Django as well as the React library.
Professional Strength:
Nicholas possesses strong leadership skills having borne the mantle in multiple clubs and societies with a proven track record of improvement during his term. Nicholas is also a driven individual with good initiative and will always commit to improving on his works.
Fun Fact:
Nicholas is an avid turtle lover and all things turtle to the point where he has been called to look similar to one several times.

Team Schedule

The team will be having a meeting every Wednesday and Saturday night as team members all free during those times aside from the regular weekly meetings during the team's tutorial class (Friday afternoon, 2 - 4 pm). The team's preferred method of communication is through online communication platforms such as Discord. These bi-weekly meetings serve to do sprint planning, team retrospectives, update each other on the progress which has been made as well as to hold discussions for any roadblocks that are encountered.

The team has also agreed to allocate a set amount of 6 hours per week to work on the assignment either individually or collectively. This ensures that the team makes cumulative progress on the assignments over the course of the semester instead of rushing the work at the end. Of course, team members are free to contribute more hours to the group assignment to do any enhancements if they deem necessary.

The workload will first be broken down into small manageable tasks which will be distributed to the team members based on a combination of multiple factors including their enthusiasm in taking up the task, their past experience which relates to the task, and the difficulty of the task. All members of the team should have an equivalent amount of task difficulty after distribution. To track the progress of the task, the team will employ the use of Trello where team members can move their tasks between statuses to reflect on the tasks' progress. Should a team member feel that they might not be able to complete their allocated tasks within the set timeframe, other team members are free to take up the task and the busy team member would have to replace the hours at a future time.

Technology Stack and Justification

Programming Languages

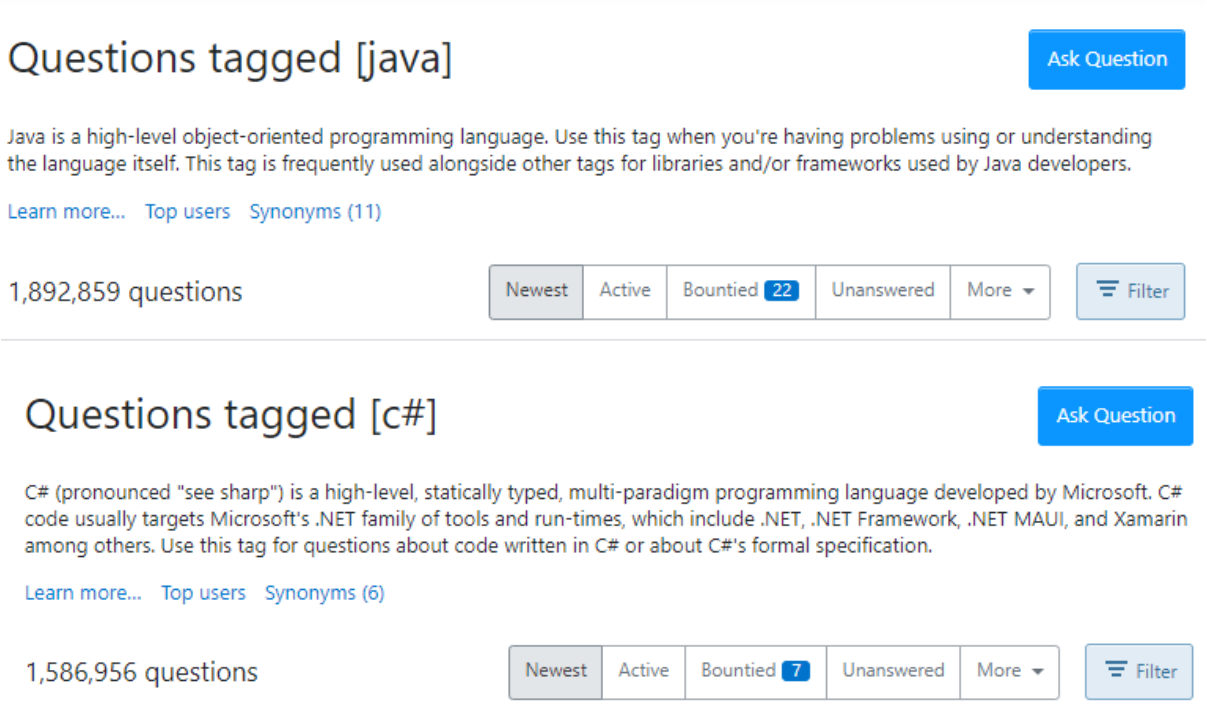
At first, we had no idea what programming languages to use so we listened to the recommendations from our lecturers and decided to compare them. Thus, we compared Java and C# based on their project requirements, community support, performance, portability, and the team's current expertise to decide on the language used for this game.

Project Requirements

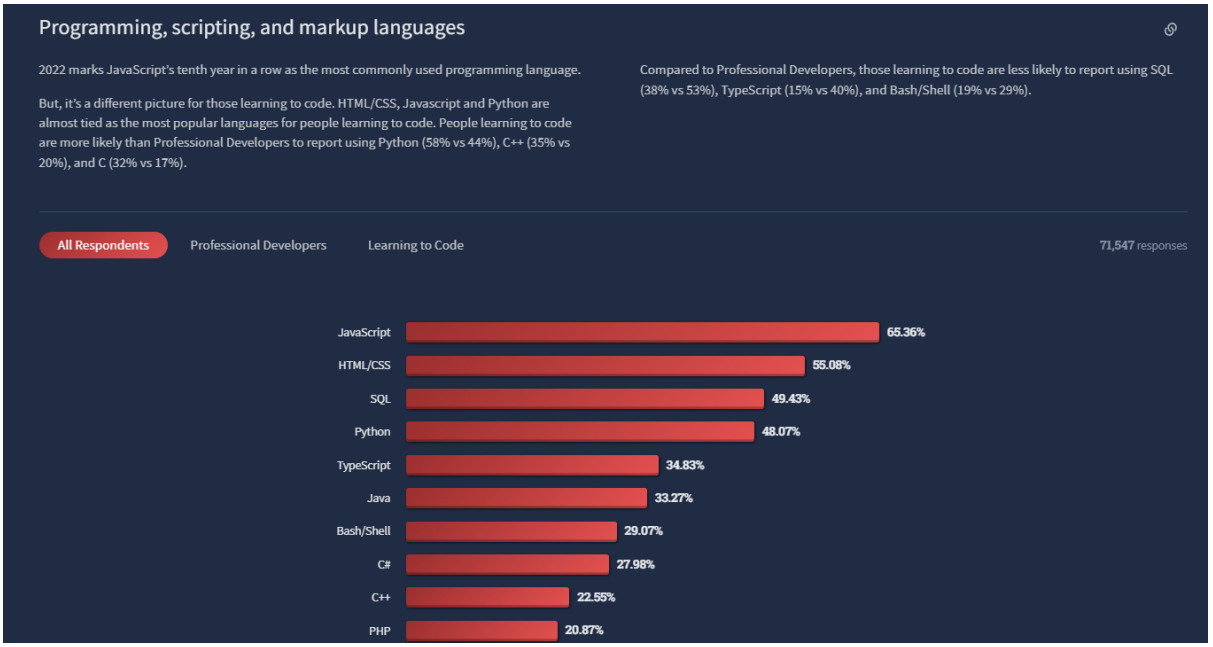
We are required to build a standalone application that has to be OOP and can run locally on a single device. While they do not mention what device, to be on the safe side, we will assume that it can run on either a mobile device or a desktop.

Community Support

Java has been around longer than C#, as it was released in 1996 compared to C# in 2002. Thus, it has more developers compared to C#. In addition, we can see that questions tagged with 'java' is more than the questions tagged with 'c#' in StackOverflow. This indicates that we will have a higher chance of finding a solution in StackOverflow about Java as some may have asked that question.



Not to mention, Stack Overflow conducted a 2022 Developer Survey and the results were clear when Java was more commonly used compared to C# with Java being 33.27% and C# at 27.94%.



Performance

C# performs faster and consumes less memory compared to Java based on the [23.03 Benchmarks Game]. It makes sense as .NET's uses JIT compiler optimizations which allows for effective code optimization while making the code more compact, making it highly productive. Not to mention, Java lacks functions such as Delete or Free, disallowing users to have direct control over garbage collection, resulting in higher memory usage.

Portability

Java can run across multiple platforms as it focuses on WORA (Write Once Run Anywhere) and does not require extra frameworks to port to other platforms. In comparison to C#, it requires the .NET Core framework to ship the code for other platforms. However, .NET Core focuses on building web applications and is not recommended for standalone applications.

Team's Current Expertise and Tutor Support

Although Jensen has experience in C# as he has created software with Unity Engine, all of us are more experienced in Java as we have taken FIT2099 previously.

Thus, we will require support if we develop our application with C# from our lecturer and tutor and will not require as much for Java.

Conclusion

Although C# is better for performance, Java is more suited for our project requirement, has more community support, can be easily ported to multiple platforms, and our team is more experienced in it. Thus, we have decided to choose Java as our programming language.

APIs

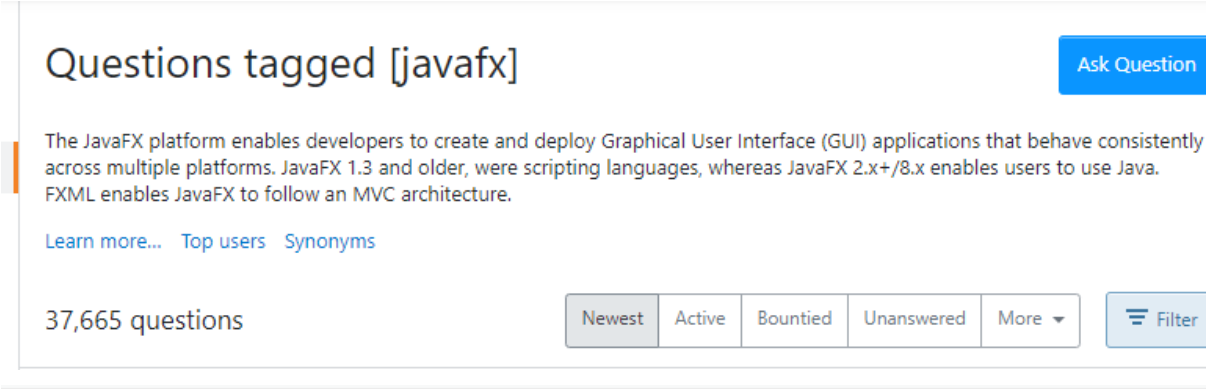
As of the current state of our plans, we have not decided on an API that we will be using. However, this is subject to change and if we decide to add things along the way, we will definitely look into APIs that will improve our work efficiency.

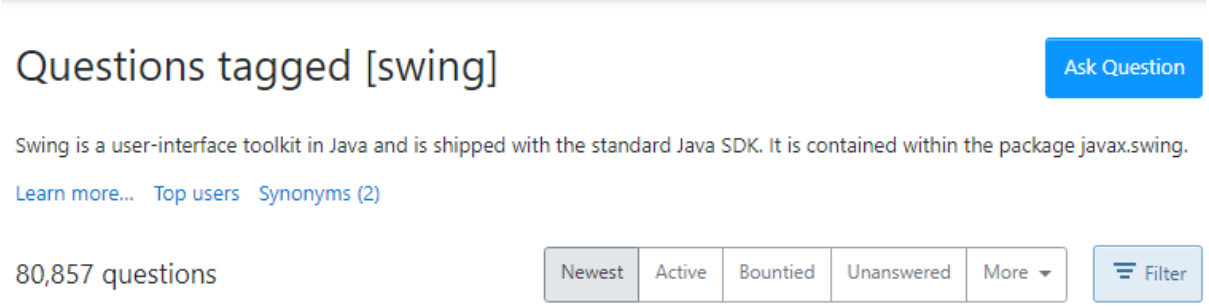
Technologies

Since we have decided to use Java, we'll be going over which GUI to use to build our standalone application. At first, we were thinking of Java Spring, as it is widely used in the industry with Hibernate. However, the framework focuses on web applications which therefore does not suit our project requirement. Thus, we decided to shift our focus to 2 main frameworks, JavaFX and Java Swing.

Community Support

Java Swing has been released since 1996 while JavaFX was released in 2008. Therefore, there will be more developers working with swing compared to JavaFX. We will first compare the available questions in JavaFX and Java Swing in StackOverflow.





It appears that JavaFX has fewer questions asked compared to swing. Thus, it will be much easier to find a possible solution for a question that we have with swing.

JavaFX vs Java Swing

Based on this article, JavaFX is a successor to Swing and provides a clear and clean architecture and features many enhancements such as styling, event management, and transitions. However, it is noted that many of the third-party libraries have not been available for JavaFX and only available in Swing.

Based on this article, it suggests that we choose Swing if we need a stable, reliable, and supported framework with a traditional look and feel. It also allows us to create complex components. On the other hand, JavaFX would be the better choice to create modern and

visually-rich desktop applications with multimedia features and animations. It also provides a modern architecture that allows for complex UIs.
Regarding GUI Builders, it exists for each of the Java libraries. Java Swing offers a GUI builder called WindowBuilder by Google while JavaFX has one called Scene Builder.

Team's Current Expertise and Tutor Support

None of us is familiar with either JavaFX or Java Swing. However, one of our tutors is familiar with JavaFx and we could ask him for his clarification if necessary.

Conclusion

After carefully considering our options, we have ultimately chosen to utilise JavaFX with Scene Builder as our GUI library and builder. The decision was not an easy one as we found both libraries to be quite similar in many aspects. However, our tutor, who has prior experience building software with JavaFX, will be able to provide us with much-needed guidance and support along the way. Not to mention, JavaFX is the successor of Swing that provides a modern look and feel to the application. In addition, JavaFX provides better support for incorporating animations, which would significantly help us as we are planning to add animations to our Nine Men's Morris game.

Final Choice of Technologies

Java with JavaFX and Scene Builder.

2. User Stories

Roles used in User Stories

Game menu:
The game menu represents the menu which appears in the first screen that the player is greeted with upon starting the game. The game menu should contain some options which are selectable by the player.

Game board:
The game board represents the Nine Mens' Morris board which is the main thing, other than the tokens, that the player will interact with during the game. The game board contains the positions that the player can place their tokens on with lines connecting these positions.

Game stage:
The game stage represents the area outside of the game board during the game. Its main purpose is to give more information to the player during the game so that the player can plan his/her strategy better.

Game system:
The game system represents the system that governs the state of the game. The game system's purpose is to prevent players from performing illegal moves, change game phases and end the game when a player wins or draws.

Game result:
The game result represents the result that would show up after the game has ended. It will provide information on who has won the game, the points earned, and allow the player to restart the game.

Game player:
The game player represents any player during the game. The game player essentially commands the flow of the game by moving the tokens around and forming mills.

AI:
The AI can play on the game board with the player instead of another player. The AI controls the placement and movement of their tokens without the input of a player.

Before Game Starts

As a game menu, I want to allow a new game to be started so that the board game can be played.

As a game menu, I want to allow the game instructions to be viewed so that new players can learn the flow of the game.

As a game menu, I want to allow different game modes to be selected so that the player can choose to play with another player or an AI.

Throughout the Game

As a game stage, I want to display how many tokens the player has left to place on the board so that the player is more informed about the decisions they make when placing the tokens.

As a game player, I want to easily differentiate between which tokens belong to me and my opponent so that I can avoid confusion in the game.

As a game stage, I want to show when it is each player's turn so that the respective player can make a move.

As a game system, I want to ensure no illegal moves are made so that a fair game can be played.

As a game board, I want to show all of the opponent's tokens that can be removed by the player when he/she has formed a mill so that the player will not remove an opponent's token that is in a mill.

As a game board, I want to show the position of the player's mill during his/her turn so that the player can view them more easily.

As a game system, I want to make sure the player has available moves so that the player has not lost the game yet due to the inability to move.

As a game system, I want to check if the player has more than 2 tokens so that the player can continue with the game.

As a game system, I want to declare that a player has lost when he/she has less than 3 tokens so that the game can come to an end.

As a game system, I want to ensure that players will be able to remove one of their opponent's tokens when they form a mill so that they can put their opponent at a disadvantage.

Game Phase 1 - Placing Tokens

As a game board, I want to start out empty at a clean state so that the first player can place his/her token at any position.

As a game player, I want to have nine tokens at the beginning of the game so that I can place them on the board.

As a game player, I want to place my token at any empty position on the board at the beginning of the game when my nine tokens have not been exhausted so that I can move them later in the game and plan out my starting strategy.

As an AI, I want to place my token based on a set of heuristics at any empty position on the board at the beginning of the game when my nine tokens have not been exhausted so that I can move them later in the game.

Game Phase 2 - Moving Tokens

As a game board, I want to show all the tokens the player can move (tokens with adjacent empty positions) during his/her turn so that the player is more informed and can decide which token to move.

As a game board, I want to show all of the positions the player can move with their selected token so that the player is more informed and can decide where to move the token.

As a game player, I want to select my movable token on the board to move it to an adjacent empty spot so that I can form a mill in this turn or the next.

As an AI, I want to select my movable token on the board based on a set of heuristics to move it to an adjacent empty spot so that I can form a mill in this turn or the next.

Game Phase 3 - Flying Tokens

As a game player, I want to move my token to any empty position on the board when I am down to my final three tokens so that I can form a mill.

As an AI, I want to move my token based on a set of heuristics to any empty position on the board when I am down to my final three tokens so that I can form a mill.

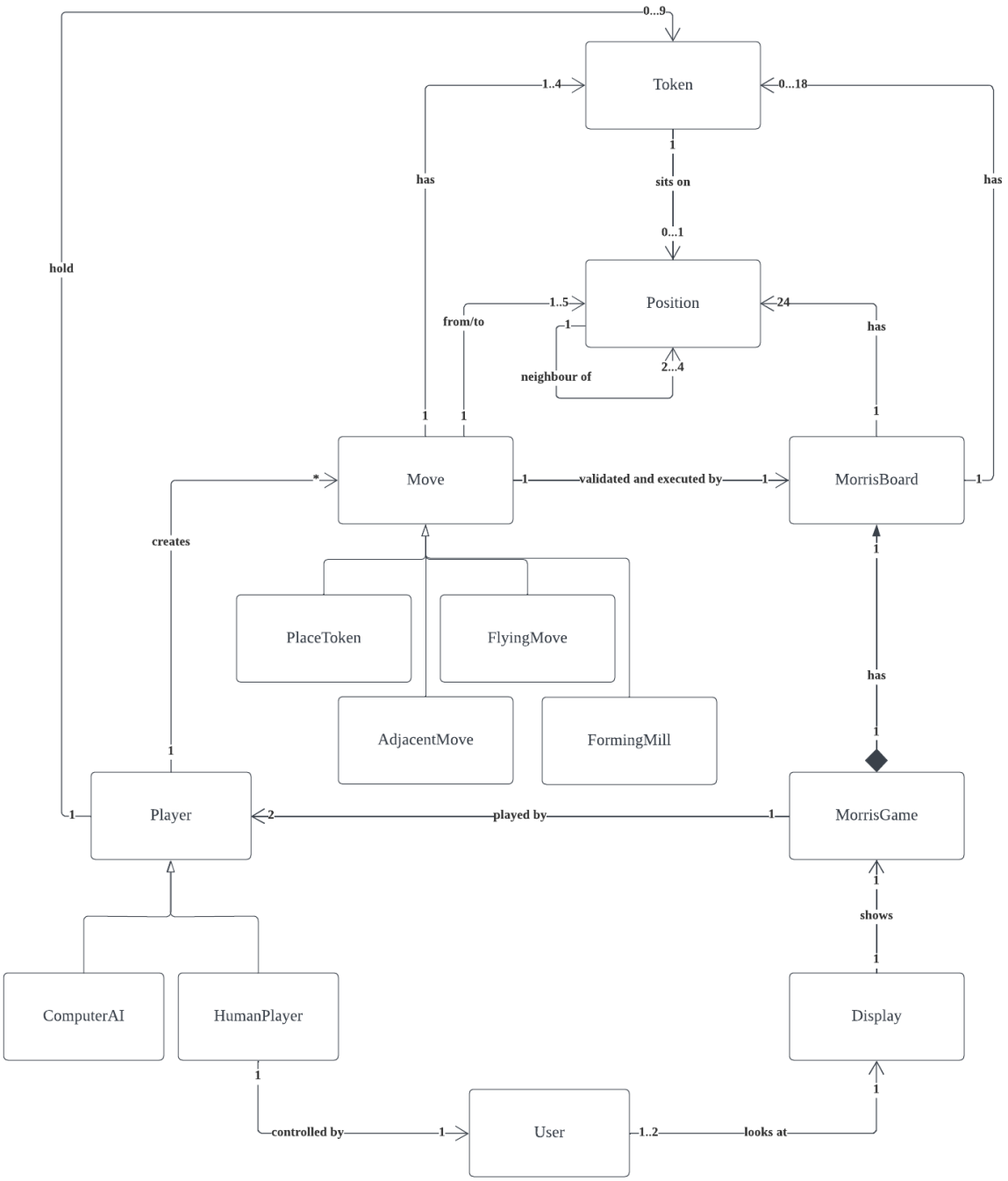
After Game Ends

As a game result, I want to show the result of the game played so that the players are informed of the outcome of the game.

As a game result, I want to allow the player to restart the game so that the player can play the game again.

3. Basic Architecture

Domain Model



Rationale for each chosen domain

Entities	Rationale
User	The User represents the real-life person playing the Morris game. This entity exists to indicate that the HumanPlayer will be played by us and that we will be looking into the display while playing it.
Display	The Display entity shows the user interface of the Morris game to the User. It is the contact point that connects the User to the underlying logic of the game.
MorrisGame	The MorrisGame entity represents a single instance of the 9 Men Morris Game. The MorrisGame entity can be restarted.
Player	The Player entity is the parent class of both the ComputerAI and the HumanPlayer entities as it contains variables and methods common to both children.
MorrisBoard	The MorrisBoard entity represents the 9 Men's Morris Game board that holds the positions, and tokens and validates all the moves. Without it, the user can perform illegal moves and may potentially break the game.
Move	The Move entity is the parent class of the PlaceToken, AdjacentMove, FlyingMove, and FormingMill entities as it contains variables and methods common to all children.
Token	The token entity represents the tokens in 9 Men’s Morris. It is important that this entity exists to demonstrate the interactions of the tokens with other entities.
Position	The Position entity represents a position on the Morris board. A single position can be unoccupied or occupied by a maximum of one token.
Computer AI	The ComputerAI entity represents a player that is controlled by the computer. Without this entity, the computer could not control the game and it would not be able to fulfil the requirement.
HumanPlayer	The HumanPlayer entity represents the player that is controlled by the human. Without this entity, the game could not be controlled by humans from playing the 9 Men’s Morris game.
PlaceToken	The PlaceToken entity represents a type of move that the player can do at the start of the game, when they still have some tokens in their hand. The purpose of this entity is to allow players to place their tokens onto the board at the start of the game so that they can move them later on.

AdjacentMove	The AdjacentMove entity represents a type of move that the player can do after placing down all of their tokens on the board and still have more than 3 tokens on the board. The purpose of this entity is to allow the player to move the token to an adjacent empty position.
FlyingMove	The FlyingMove entity represents a type of move that the player can make when they are only left with 3 tokens left on the board, where they would move any of their tokens to any empty position. Without this entity, the player would not be able to perform this move and it will make the game significantly harder to play.
FormingMill	The FormingMill entity represents a type of move that the player can make when they have three tokens adjacent to each other on the Morris board either vertically or horizontally enabling them to remove one of their opponent's tokens which is not in a mill. This entity is the key winning strategy of the game.

Rationale for each Domain's Relationship

Entities	Relationship	Justification
User	Association with Display	The user will be using one display to look at the current game that they are playing.
Display	Association with MorrisGame	The display shows the user the current MorrisGame that is being played.
MorrisGame	Association with Player	A game of 9 Men's Morris can only be played when there are exactly two players.
	Composition with MorrisBoard	The Morris Board only exists if the Morris Game exists.
Player	Association with Token	A player will always start with 9 tokens at the start of the game and will have 0 tokens once the player has placed all of them on the board.
	Association with Move	A player can create multiple moves throughout the entire game to form mills and win the game.
MorrisBoard	Association with Token	A 9 Men's Morris game can have 0 to 18 tokens on the board, where it has 0 tokens during the start of the game and 18 tokens when both players have placed all of their tokens on the board.
	Association with Position	A 9 Men's Morris game will always contain 24 positions.
Move	Association with MorrisBoard	Every move that is being created by the player needs to be validated and executed by the MorrisBoard to change the game into a different state.
	Association with Token	A player needs to select a minimum of one token or multiple tokens depending on the specific type of move to play their turn.

	Association with Position	A player needs to select a minimum of one position or multiple positions depending on their specific type of move to play their turn.
Token	Association with Position	Each position in the 9 Men's Morris game has 2 to 4 neighbouring positions. The token needs to know the position that it is located in to identify whether it is part of a mill.
Position	Association with Position	The position needs to know its neighbouring position to allow tokens to perform adjacent moves or identifying mills.
Computer AI	Generalisation of Player	A ComputerAI is a player that can create a move and play the 9 Men's Morris game except for the moves that it makes are generated by the computer instead of retrieving it from the player.
HumanPlayer	Generalisation of Player	A HumanPlayer is a player that can also create a move and play the 9 Men's Morris Game but the moves are retrieved from the player instead.
	Association with User	A HumanPlayer must be controlled by a user because it needs to be controlled by a human.
PlaceToken	Generalisation of Move	The PlaceToken entity is a type of move that allows the player to place their token in an empty position on the board.
AdjacentMove	Generalisation of Move	The AdjacentMove entity is a type of move that allows the player to move one of their tokens on the board to an adjacent empty position.
FlyingMove	Generalisation of Move	The FlyingMove entity is a type of move that allows the player to move one of their tokens on the board to

		any empty position when the player only has 3 tokens left on the board.
FormingMill	Generalisation of Move	The FormingMill entity is a type of move that allows the player to form a mill when three tokens are adjacent to each other horizontally/vertically and then remove a token from the opponent player which is not in a mill.

Design Choices of Domain Modelling

Design Choice of Modelling - Command Design Pattern

We have based our domain model around the Command Design Pattern. The idea behind this design pattern is that the “Invokers” (Senders) do not send requests directly to the underlying business logic but instead, these requests go through another layer called the “Command” layer. This layer packages these requests into consistent, encapsulated, objects and handles all the details of execution. In our domain model, the Player entity is the “Invoker” while the Move entity is the “Command”. The Move class will ensure that all of its children classes implement a common method which can be used to execute the command. The MorrisBoard will act as the “Receiver” of these commands that validates and executes all of the functionality packaged within them.

Additionally, by employing the Command Design Pattern, we are abiding by two principles of SOLID namely the Single Responsibility Principle and the Open-Closed Principle. Having the Command layer in between allows the decoupling of classes that invoke and perform operations while ensuring that the details of each command executed are hidden, achieving the goal of Separation of Concerns. Aside from that, new code and features can also be added easily to this design pattern without breaking the existing code already, making this design pattern extensible. For instance, a possible new feature to be added includes the undo/redo functionality.

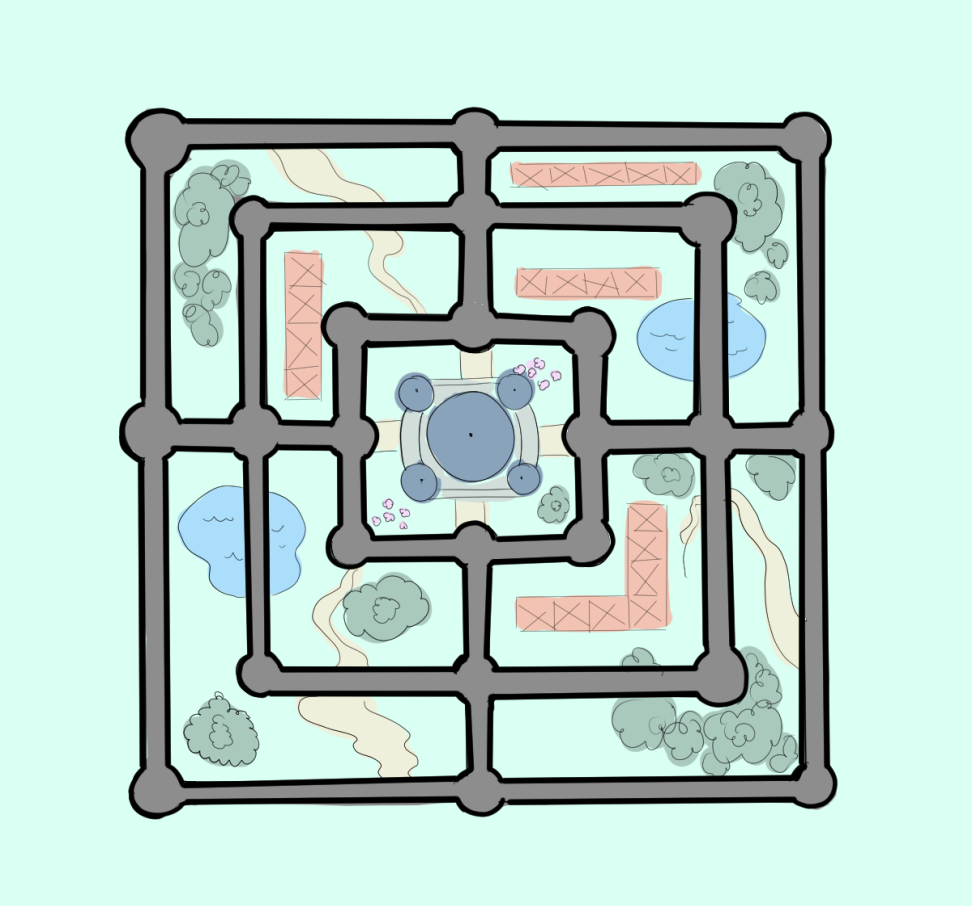
Discarded Alternative Design Choice - Observer Design Pattern

As an alternative, we have also looked into the Observer Design Pattern. On the surface level, it might seem attractive as player entities involved can subscribe or unsubscribe to the board entity to be notified of the latest state of the game. However, this means that there would be a need for additional side logic to handle the subscription and unsubscription of services because if not, subscribers would be notified about things that they do not need to know, violating the Separation of concerns. Additionally, the Observer Design Pattern does not allow for subscribers to send commands over to the logic, which would disallow players from playing the game. However, a bright side to this design pattern is that it would open up the opportunity to implement a spectator feature if desired.

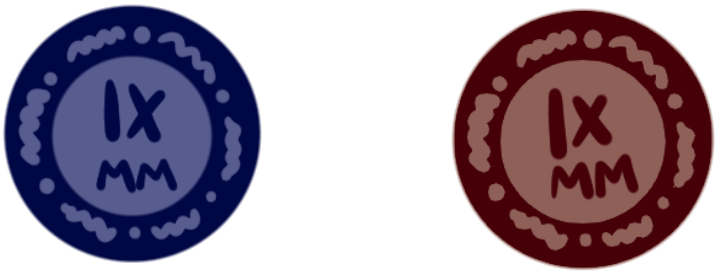
Assumptions

With the command design model, we assume that the future special moves will be placed below the Move entity and be validated by the MorrisBoard. While modelling the domain, we also modelled it for the 9 Men’s Morris and not for other versions. However, the Morris Board entity can be designed to have generalisations of other boards such as 6 Men’s Morris for extension.

4. Basic UI Design



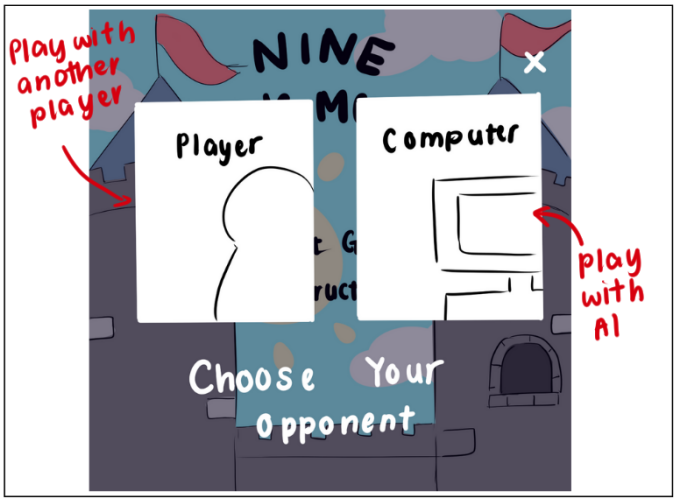
Board Design



Token Designs

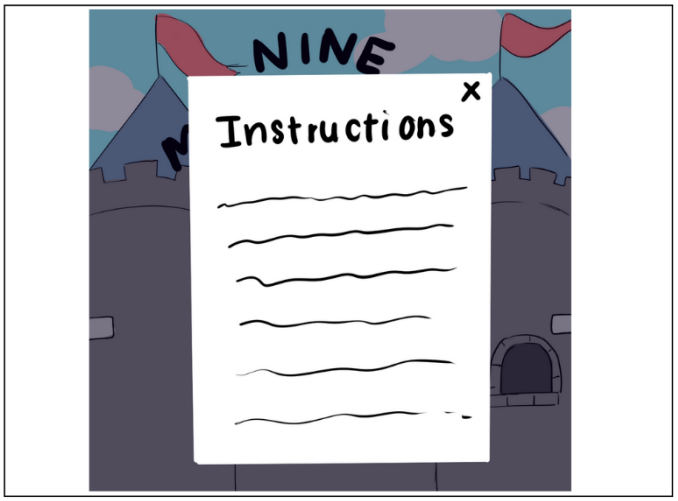
Frame: Title Screen

The first screen that the player sees when entering the game.



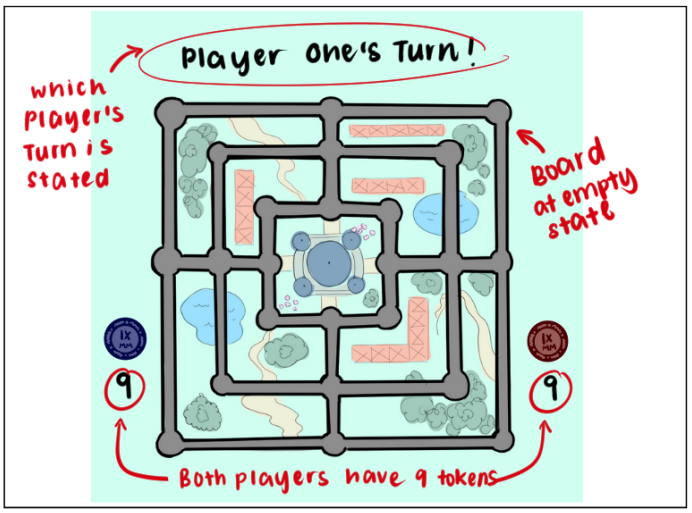
Frame: choosing an opponent

When player selects "start game" on the title screen, player will be prompted to choose their type of opponent.



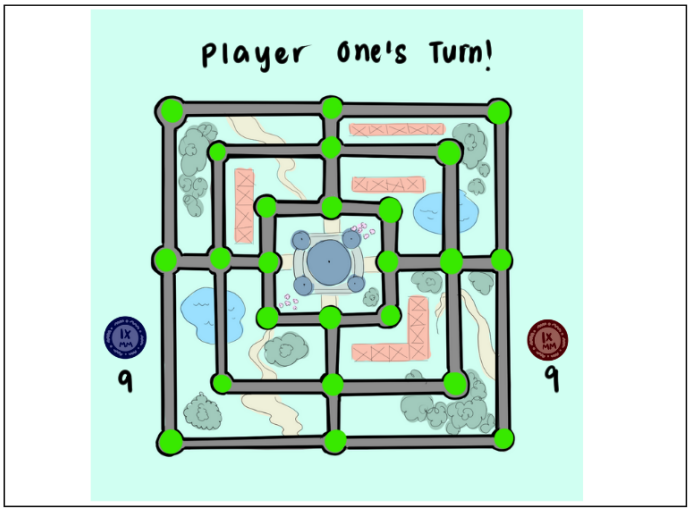
Frame: Viewing Instructions

when player selects "instructions" on the title screen, player can read up the instructions of the game



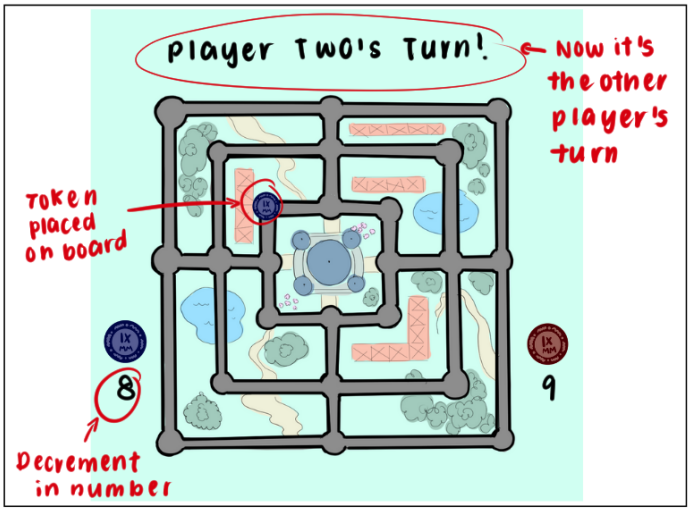
Frame: Initial Board

After selecting the type of opponent, player will be led to the Nine men's Morris Board at its initial state. Both players should have nine pieces in their inventory.



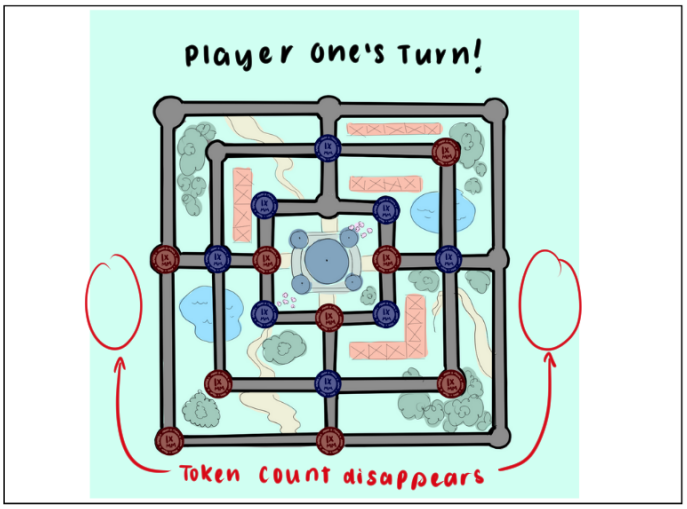
Frame: Game Board showing all possible locations to place token

During the placing phase at the start of the game, the player will be shown all possible (empty) locations the player can place their token at. These locations are visually indicated.



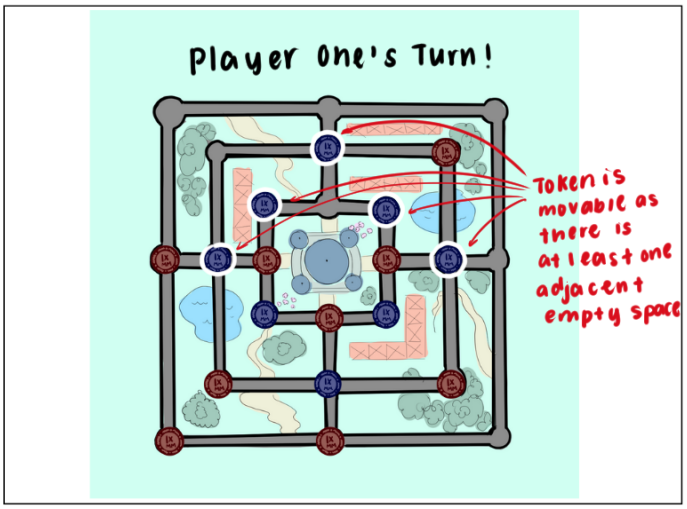
Frame: Token has been Placed.

Upon selecting a position, a token from the player's inventory will be placed, decrementing the player's token count by one. The turn now belongs to the other player.



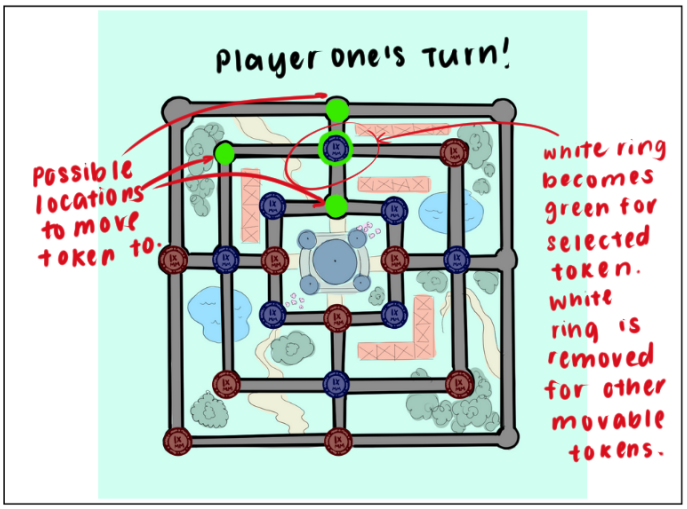
Frame: Both players have placed down all of their tokens

Placing phase ends when all players have exhausted the tokens in their inventory. Token count disappears and the game transitions to moving phase.



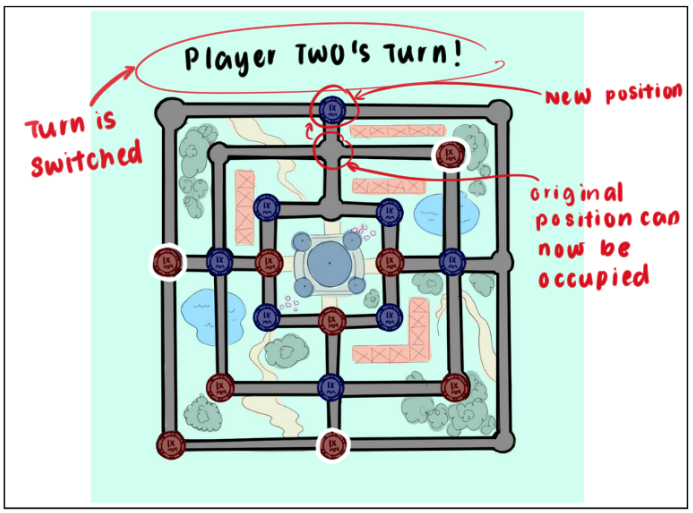
Frame: Game board showing all movable tokens during the player's turn

During the moving phase, at the start of each player's turn, their movable tokens will be visually indicated on the board.



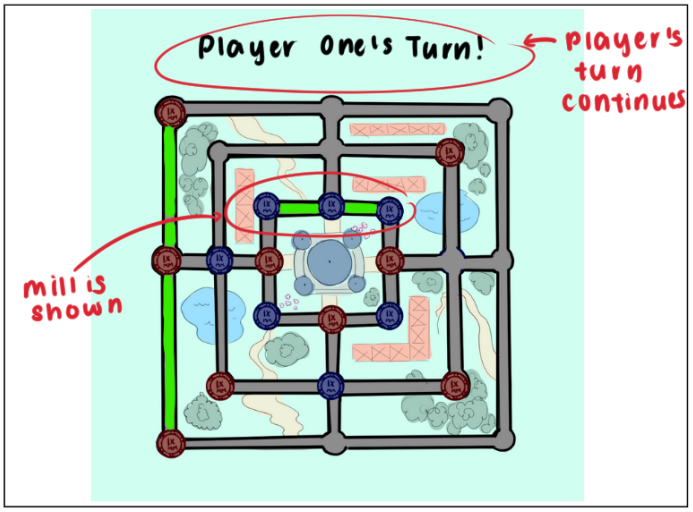
Frame: Game board showing all possible positions to move token after selecting token to move

Upon selecting the desired token to move, the selected token and its possible positions to move to will be visually indicated on the screen.



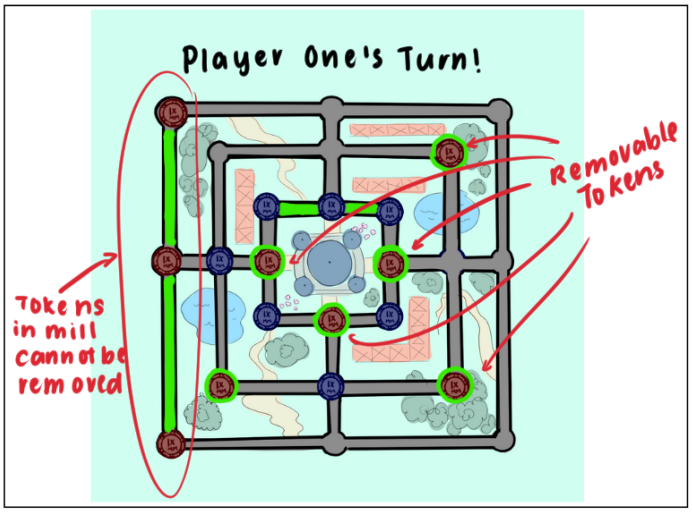
Frame: Token has been moved.

The player's token is now at its new position and the player's turn ends as no mills are formed. It is now the other player's turn.



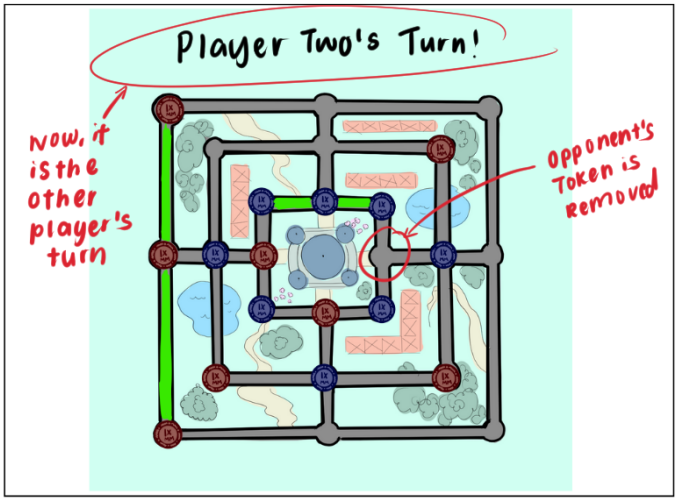
Frame: Forming a mill.

If the player forms a mill (3 of the player's tokens are adjacent in a horizontal/vertical line), the player's turn continues and transitions to removing phase. The mill is also visually indicated. Forming a mill can be done in placing phase as well.



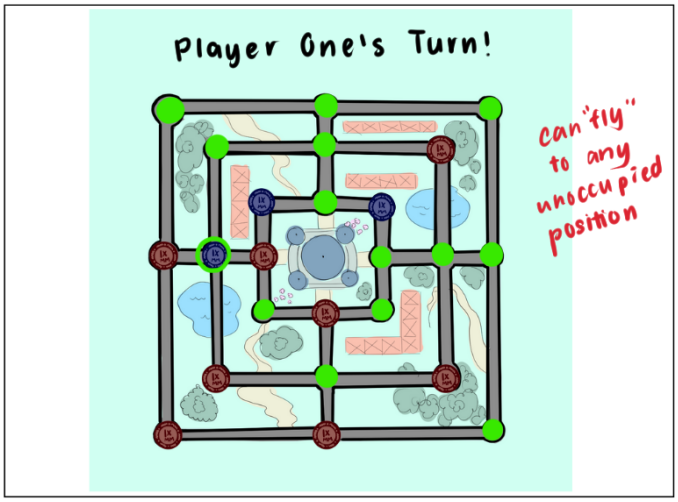
Frame: Game board showing all tokens which can be removed from opponent during player's turn.

The opponent's tokens which the player can remove are then shown on screen. Tokens that are in a mill cannot be removed. Removing a token can be done in placing phase as well.



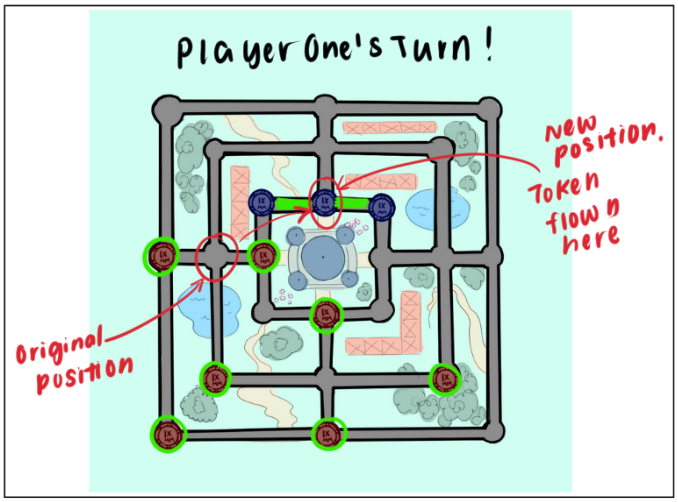
Frame: One of the opponent's token has been removed.

Upon selecting the opponent's token, the token will be removed as part of the board. The vacant space is now free to be occupied by other tokens and it is now the other player's turn.



Frame: Once down to final 3 tokens, after selecting token to move, the player can select any position on the board for the token to "fly" to.

Flying phase occurs when the player has 3 tokens. The player is free to move any of his/her token to any empty position on the board.



Frame: Token has "flown"!

Token has moved to new position. same rules apply in the game when moving the token normally (i.e forming a mill)

Frame: Results of the game

when a player has less than three tokens left or cannot move any of his/her tokens, they lose as they cannot form a mill. The other player is declared the winner. The player can then choose to restart the game or return to the title screen.

5. References

Stack Overflow Developer Survey 2022. (2022). Stack Overflow. https://survey.stackoverflow.co/2022/#section-most-popular-technologies-programming-scripting-and-markup-languages

C# .NET vs Java - Which programs are fastest? (n.d.). https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/csharp.html

Why, Where, and How JavaFX Makes Sense. (2013, March). https://www.oracle.com/technical-resources/articles/java/casa.html

A.Kumar (2023, March 19). *The Battle of JavaFX Vs Java Swing: Which One Is Best For You?* DevDojo. https://devdojo.com/anvesh_kumar/the-battle-of-javafx-vs-java-swing-which-one-is-best-for-you

Refactoring Guru (2023, January 1). *Command*. https://refactoring.guru/design-patterns/command

Refactoring.Guru. (2023b, January 1). *Observer*. https://refactoring.guru/design-patterns/observer