Project Specifications

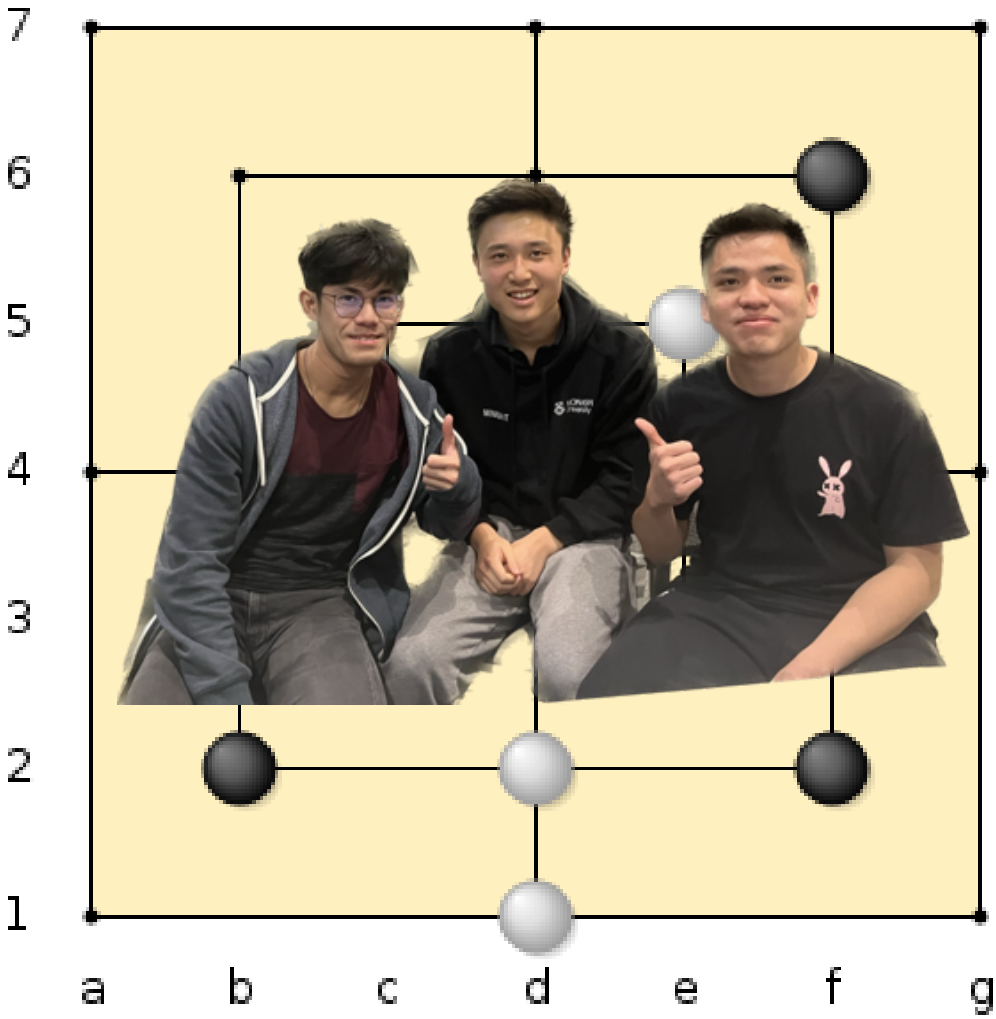
Basic Architecture and Design Rationale

Prepared by 3 Men's Morris
Jer (Arthur) Lin, Quoc (Harry) Han and Liangdi Wang

Contents

1. Team Formation

Team Name: 3 Men's Morris



1.1. Team membership

Jer (Arthur) Lin

Contact details	Strengths	Fun facts
jlin0074@student.monash.edu	DevSecOps Basic GIT skills CI/CD pipelines Web development	Can solve a Rubik's cube in under 25 seconds Made a mobile game called "Sassy cat" on the Play Store

Quoc (Harry) Han

Contact details	Strengths	Fun facts
hhan0014@student.monash.edu	Network security Networking	Like coding mobile apps (either on Android or IOS).

Liangdi Wang

Contact details	Strengths	Fun facts
lwan0141@student.monash.edu	Web development Deploying full stack to Cloud	Like to build PCs - has built 15-20 PCs for friends, family, work Can solve a Rubik's cube in under 25 seconds

1.2. Team schedule

Regular meeting time at 7:30PM every Saturday if required

Meeting time	Duration
25/03/2023, 7:30PM - 8:00PM	0.5 hours

This meeting will be used to discuss progress, distribute workload, and make any necessary decisions.

We will also have a weekly check-in: Every Thursday at 6:00 PM. This will be a brief check-in to see how everyone is progressing and to discuss any issues that may have arisen.

Regular work schedule:

- Work is done throughout the week whenever team members have time.
- However, each team member is expected to contribute at least 3 hours of work per week to the project.
- Team members are expected to contribute an even amount, and will contribute more hours per week when necessary (more tasks need to be completed)
- Team members will collaborate via text or voice channels in our Discord server when we are working together.

Workload distribution:

- Workload is distributed during the weekly team meetings.
- Each team member is responsible for completing their assigned tasks by the agreed-upon deadline.
- If a team member is unable to complete their tasks or if any changes are to be made, they should communicate this to the rest of the team as soon as possible.
- Work done is tracked in the team GitLab repository contribution log located in the “wikis” folder.

1.3. Technology stack

Programming languages and technologies:

- TypeScript: TypeScript is a superset of JavaScript that is strongly typed, adds static typing, interfaces, and other features that have good support for object oriented design patterns. With TypeScript, we can catch type errors earlier in the development process, which allows for easier development for a complex game like 9 Men’s Morris. While none of us have used it before, it is a popular language with lots of online resources to help us learn and use it.
- React: React is a component based frontend web framework that supports design principles of reusability and reducing repeated code. This is well-suited to building reusable components for different parts of the game board and UI. React apps can also be run and built locally which supports the requirement of “The application you will eventually develop must be implemented as a standalone application that is able to run locally on a single device and does not require separate server-side code to be written.”

Mapping to team's current expertise:

- We have experience in web development, which should help us pick up TypeScript relatively easily.
- One team member has used React to develop a full-stack web application, so this experience will be useful in developing the UI.

Anticipated support from tutors:

- None of us have used TypeScript before, so we might need some help from tutors in learning and implementing it effectively.

Justification of final choice:

- TypeScript helps in the development of a complex game like Nine Men's Morris since it has good support for object oriented design patterns and strong typing.
- React is a component-based frontend framework with an emphasis on creating reusable components, which should help simplify the UI development.
- Together, they allow us to easily build a user interface with reusable components while using object-oriented design patterns to code the game logic.
- Our team has experience with web development so this technology stack will be suited to our strengths.

2. User Stories

2.1. Basic requirements

1. As a player, I want to see a visual representation of the game with the board and the pieces, so that I can play accordingly.
Acceptance criteria:
- When the game starts, the board is visible
- As pieces are places, they show up where they are placed

2. As a player, I want to be able to start the game, so that I can commence playing the game.
Acceptance criteria:
- When the player clicks on the start, create a board Object
- Initialise two players, 9 pieces of each colour and all the position coordinates
- Prompt the player with white pieces to start

3. As a player, I want to be able to place my pieces on empty intersections, so that I can try to form a mill.
Acceptance criteria:
- When a player has pieces and it is their turn, they can click on a valid square
- One of their pieces will be placed in that square, and they will have 1 fewer pieces they can place

4. As a player, I want to be able to form a mill, so that I eliminate the opponent's piece.
Acceptance criteria:
- Every time a player places a piece, iterate through all the positions of the pieces to see if any new mill is formed
- Every time a piece is eliminated, iterate through all the positions of the pieces to see if any existing mills are broken
- Everytime a player moves a piece, iterate through all the positions of the pieces to see if any existing mills are broken

5. As a player, I want to be able to remove my opponent's piece, so that I can reduce my opponent's resources.
Acceptance criteria:
- When a player forms a mill, prompt the player to delete an opponent piece
- If a piece is part of a mill, make it indestructible
- However, if all opponent pieces are in a mill, the player will have the option to delete any pieces, even if it is part of a mill
- Make all the removable pieces light up, so the player can choose one to remove

6. As a player, I would like to be able to move my pieces when I run out of pieces, so that I can continue to form mills.
Acceptance criteria:

- At the start of a player's turn, check if the player has any available pieces
- If the player does not have any pieces left, iterate through all the pieces and search for all the movable pieces
- All the movable pieces will light up, giving the player an option to pick which one to move
- Player then click on the spot they wish to move the piece to

7. As a player, I want to be able to fly one of my pieces to any empty position on the board if I have only three pieces left, so that I can continue playing.
Acceptance criteria:

- At the start of a player's turn, check if the player has three or less pieces left
- If so, they can move it to any position on the board

8. As a player, I want to be alerted if I try to move to an invalid position or if I try to move my opponent's piece, so that I know it is an invalid action.
Acceptance criteria:

- If a player clicks on an invalid position, tries to move the opponent's piece, or does any illegal action, an error message is displayed.

9. As a player, I want to be able to see whose turn it is, so that I know when I can make a move.
Acceptance criteria:

- Players can see text on the screen that shows whose turn it is.
- The text updates to show the other player's turn after a move has been made.

10. As a player, I want to be able to see what action I need to take (placing, moving and flying, or removing), so that I can make the right move.
Acceptance criteria:

- The player can see a message or indicator on the screen that tells them what action they need to take.
- The message or indicator updates to show the next action after a move has been made.

11. As a player, I want to be able to see my resources, so that I can plan my moves ahead.
Acceptance criteria:

- Display the number of pieces each player has not placed down yet
- Clearly indicate which player should make moves for the turn
- Provide instruction on the current action (eg. place your piece, click on the piece to remove etc)

12. As a player, I want the game to end, so that I can stop playing the game.
Acceptance criteria:
- The game ends when a player has fewer than 3 pieces.
- Ending the game should display the winner or a draw message, and prevent further moves from being made.

13. As a player, I want to start a new game, so that I can keep playing.
Acceptance criteria:
- There is an option to play again after a game has ended
- The board and game state should all be reset
- Players can continue playing

14. As a player, I want a message to be displayed when the game ends, so I can know the outcome of the game.
Acceptance criteria:
- Once the game ends, a message stating the outcome of the game is displayed
- It announces the winner
- Players cannot make any moves

2.2. Advanced requirements

15. As a player, I want to be able to play against a computer, so that I can play without another person present.
Acceptance criteria:
- There is an option to play vs computer on the starting screen
- The computer should make a random move among all currently valid moves
- The computer can only make a legal move
- The computer's move is displayed on the screen
- The game continues with the player's turn after the computer's move

16. As a player, I want the computer to make more strategic moves based on heuristics, so that I can face a challenge.
Acceptance criteria:
- The computer should evaluate potential moves based on the game state and select a move that maximises its chances of winning.

2.3. Extra functionality

17. As a game, I want each player to have a fair amount of time each turn so that the game can be finished in a good time manner.
Acceptance criteria:
- Display the time that each player still has left for making a move each turn.
- Warn players when their playable time each turn is nearly running out.
- Randomly place a piece on the board on behalf of the player when their time is run out, but this placement must obey the rules.

18. As a beginner player, I want to see the game's rules throughout the game so that I can look back at the rules when needed.
Acceptance criteria:
- Has a section/component storing rules displayed in the game's window.
- Ensure that the section/component storing rules must be presented properly that won't affect the user's experience (etc. block the board vision when displaying rules)
- Ensure that the rules are easy to understand.
- Rules should be translated in different languages.

19. As a beginner player, I want to see the valid positions to place my man on the board so that I won't break the rules accidentally.
Acceptance criteria:
- Display the valid positions in green colour.
- Allow players to place their men (points) in the position highlighted in green
- Stop players attempting to place their men (points) in the poistion unhighlighted in green.

20. As a game developer, I want to save the player's moves during the game so that I can improve the game's experience by analysing these logs.
Acceptance criteria:
- Has a log file that is used for saving all the player's moves.
- Has a clear structure of information about the player's move.
- Ask the player's permission before saving their moves.

21. As an expert player, I want to see my rank in the game so that I can know how good I am in the game during a specific time.
Acceptance criteria:
- Has a rank system in the game.
- Display clearly the player's rank over the total number of players in the game.
- Give players the artefact that reflects their rank so that they can know how far they are to achieve their desired rank.

3. Basic Architecture

The Nine Men Morris (9MM) is an ancient strategy board game played by two players. Each player will have nine men (points) to place on a board of 3 concentric squares connected by lines. Each square will have eight placeable positions thus adding up to 24 points on the board. Since the origins of 9MM are ambiguous, with no legitimate references to the rules, 9MM has many different versions. However, the game's nature (stated at the beginning) is still the same between these versions. Therefore, our group has come up with the domain model that contains standard requirements of 9MM along with additional advanced requirements.

Chosen advanced requirement:
“c. A single player may play against the computer, where the computer will randomly play a move among all of the currently valid moves for the computer, or any other set of heuristics of your choice.”

3.1. Domains

- Display: This domain is used for displaying a visual representation of the game as a user interface.
- Game: The existence of this domain is used for defining the game set up and the game’s flow. It represents a single instance of a 9 Men’s Morris game.
- Board: The existence of this domain is used for generating the environment where the players can place their pieces on. Besides that, the Board domain can be used to keep track of a piece's movement.
- Player: The existence of this domain is used for generating the player for the game. Player has a colour (white/black) and can keep track of the pieces it has on the board, and pieces that have been removed.
- Piece: This domain represents a single piece in the game, which has a colour and a position on the board.
- Move: This domain represents a single move in the game, which has a starting position, ending position, and a piece that is being moved. There are two types of move:
    - Place: This move allows a piece to be moved into any position that hasn’t been occupied by any piece.
    - RegularMove: This move allows a piece to be moved to their adjacent position.
- Position: This domain represents a position on the board where a piece could be placed. It helps to represent the physical layout of the board and could be useful for tracking the state of each position and keeping track of adjacent positions.
- RuleChecker: This domain is responsible for checking if a move is valid according to the rules of the game. It could handle checking if the piece is allowed to move to the target position and whether that creates a mill.
- Computer: is generalised by the Player domain since computers will have 9 pieces and perform the legal moves same as the player throughout the game. However, the

difference between these two domains is that the player can restart/surrender/quit during the game while the computer cannot do it.

3.2. Relationships

- A game has a board, 2 players (or a player and a computer), and a rule checker.
- A board has 24 positions and 18 pieces.
- A piece has a colour and either a position or none.
- A move has a piece, a starting position, and an ending position.
- A move can affect 1 piece (the piece being moved) or 2 pieces (if a piece moves to a position that forms a mill, one of the opponent pieces will be removed from the board).
- Since each player has to place all their 9 pieces on the board, the minimum number of checks the Rule checker will do is 18.
- A rule checker uses the state of positions on the board to check if the move about to be performed is valid. For example: If a player wants to move their piece to one of the adjacent positions, a rule checker will first check the state of those and then highlight them in green if they are free to be occupied.
- Since the rule checker uses the state of positions to validate the player's RegularMove, each position needs to keep track of their adjacent positions.

3.3. Design choices

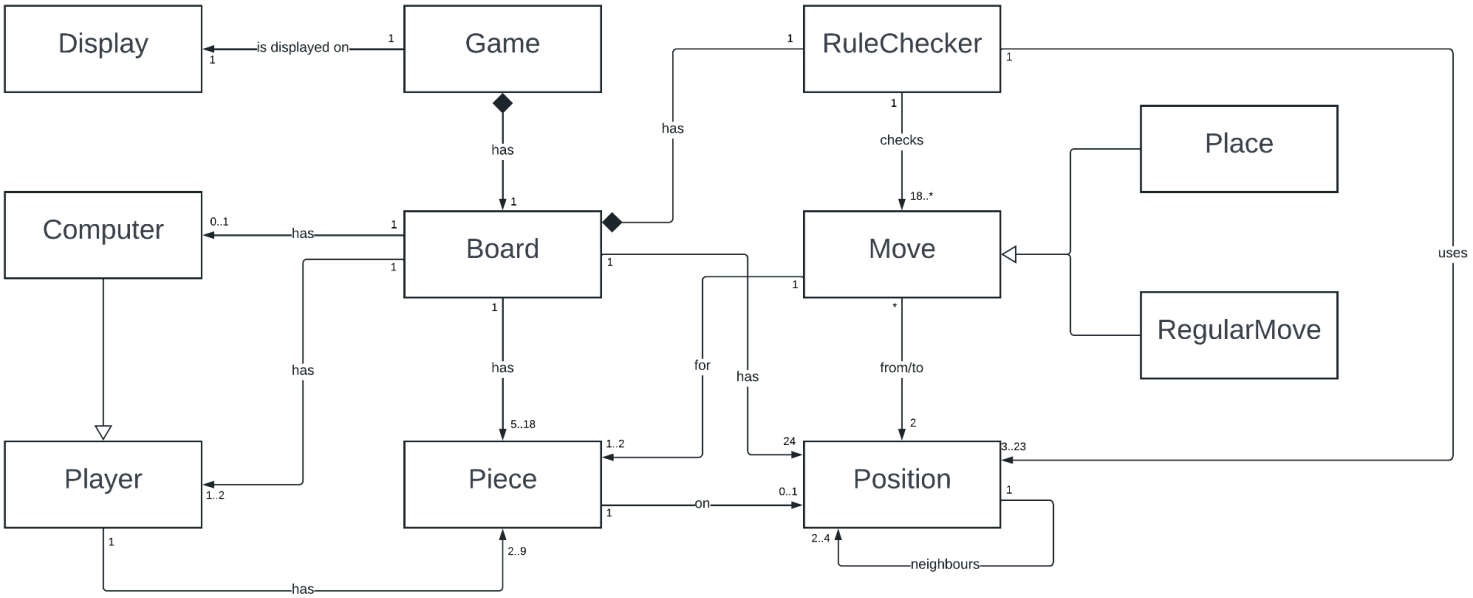
- The game is the central entity in the model, since it represents the game being played and contains all of the other entities.
- Players are modelled as separate entities, since they each have their own set of pieces and take turns making moves.
- Pieces are modelled as entities separate from positions, since they have additional properties (colour) and can be removed from the board.
- Positions are modelled as entities separate from the board to help track the state of each position and of adjacent positions.
- Using a separate RuleChecker class to encapsulate the game rules and to make them easily configurable if needed.
- Using a Move class to encapsulate the starting and ending positions of a piece, which makes it easier to pass the move data between different parts of the game.

3.4. Assumptions

- The game is played with two players only.
- The board has exactly 24 positions and no more.
- Players take turns making moves.
- The game ends when one player has fewer than three pieces or no legal moves.
- The game rules follow the standard Nine Men's Morris rules, with no additional variations or customizations.

3.5. Domain model diagram

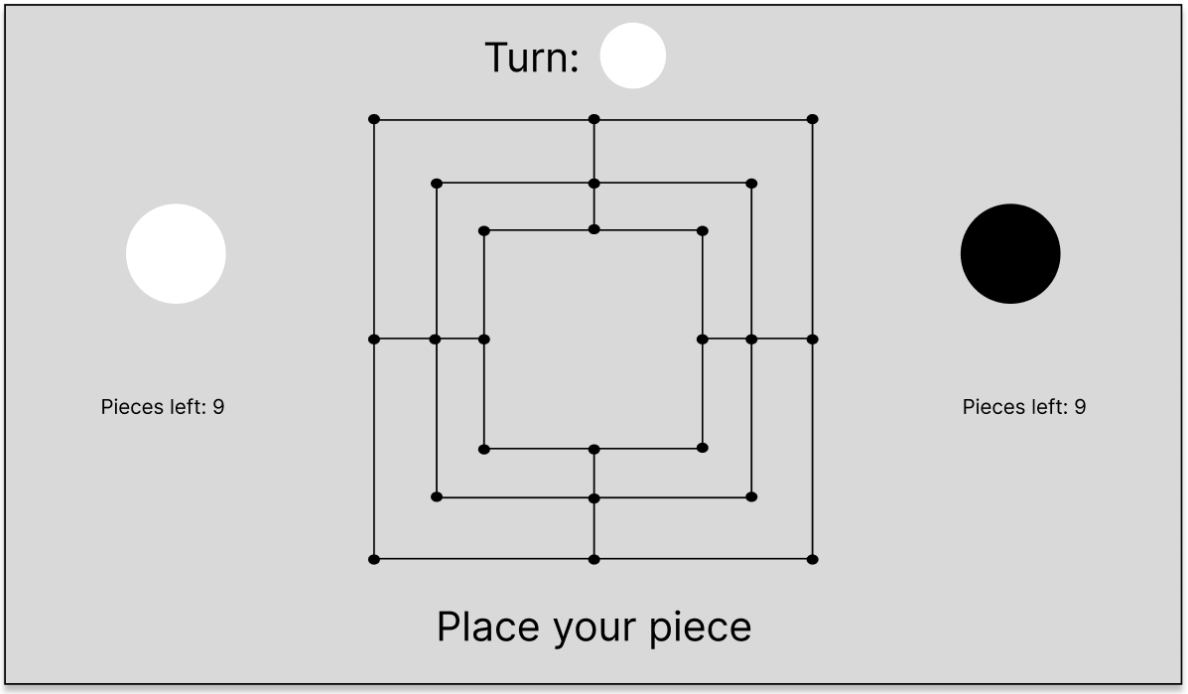
LucidChart: [link to domain model diagram]



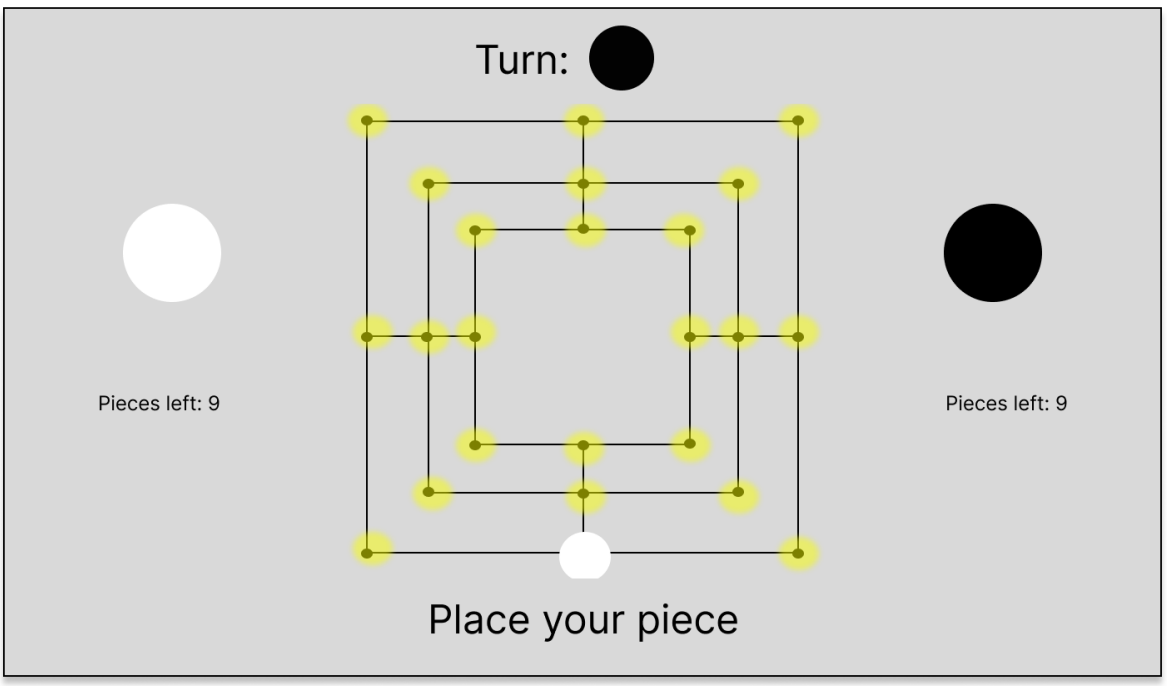
4. Basic UI Design

Figma: [link to prototype]

4.1. Initial board

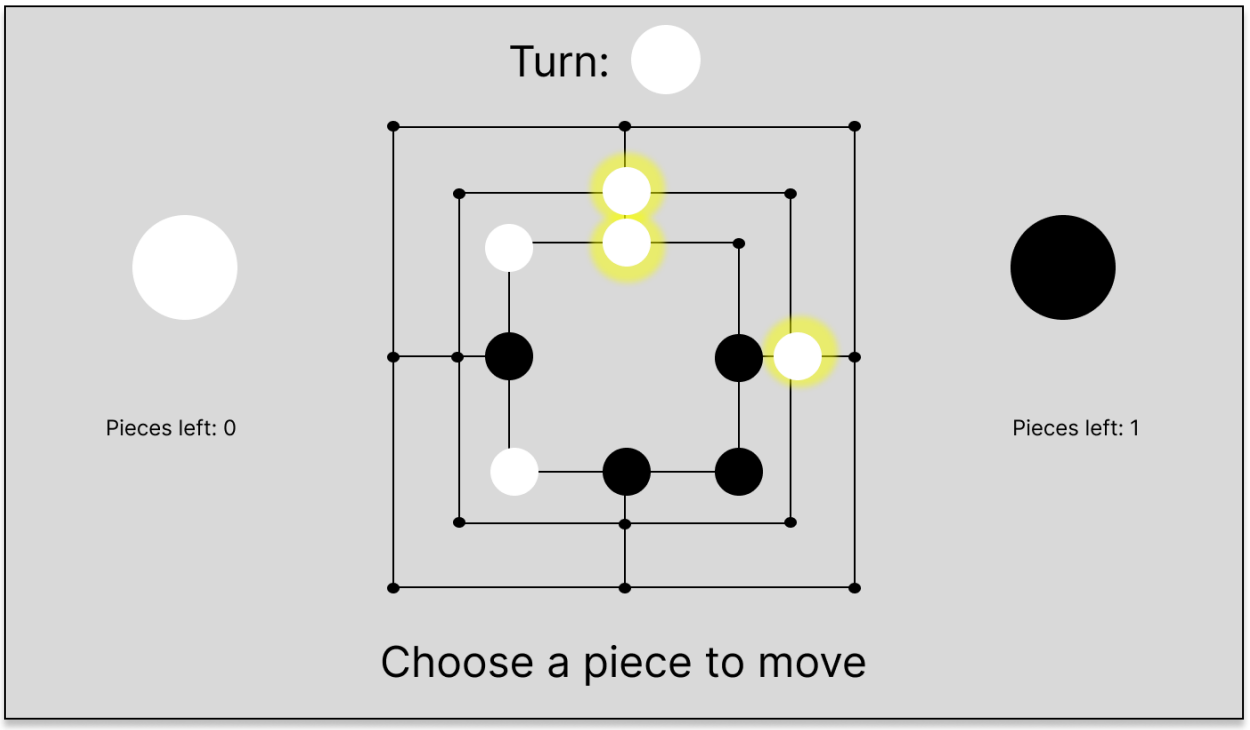


4.2. Placing tokens

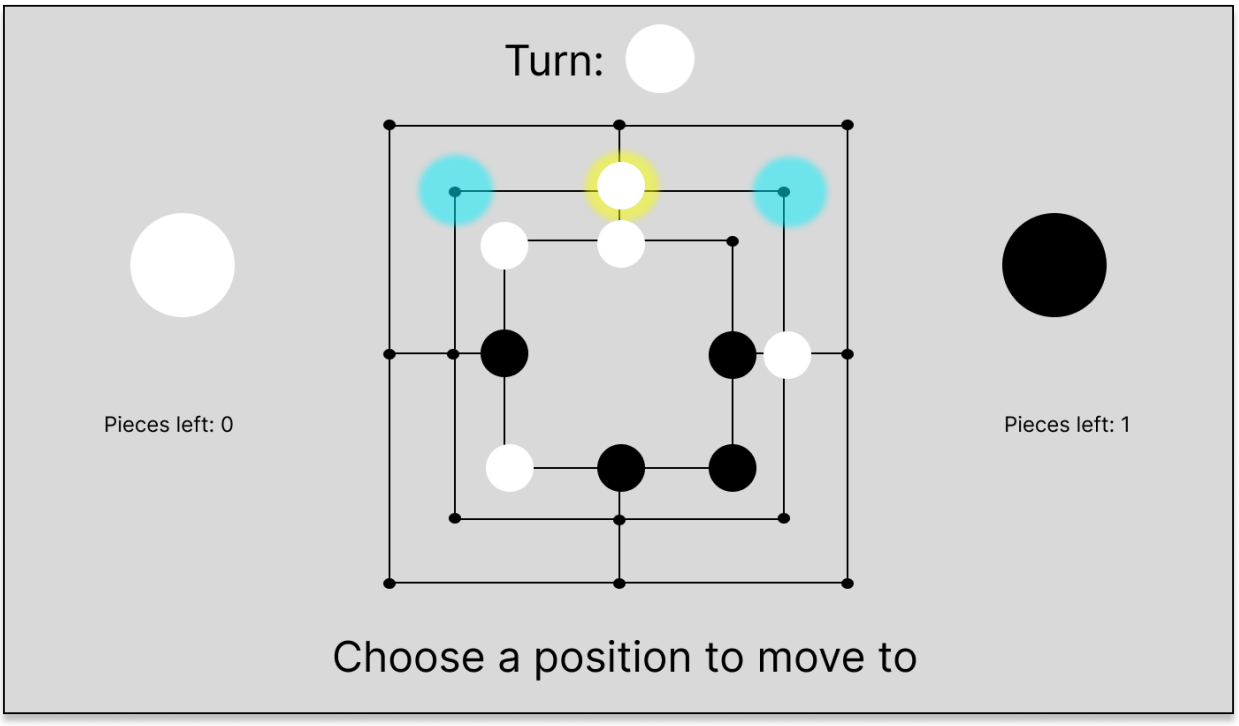


Available positions will glow up to guide the player with placements

4.3. Moving tokens

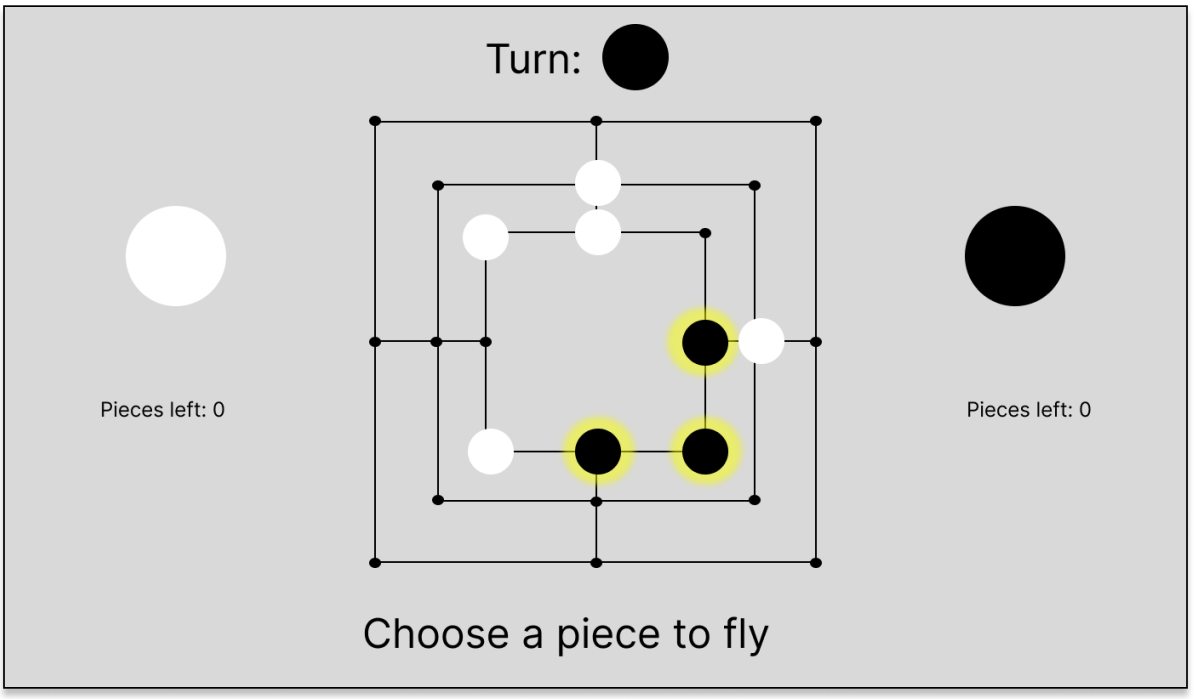


All the pieces that can be moved will glow up.

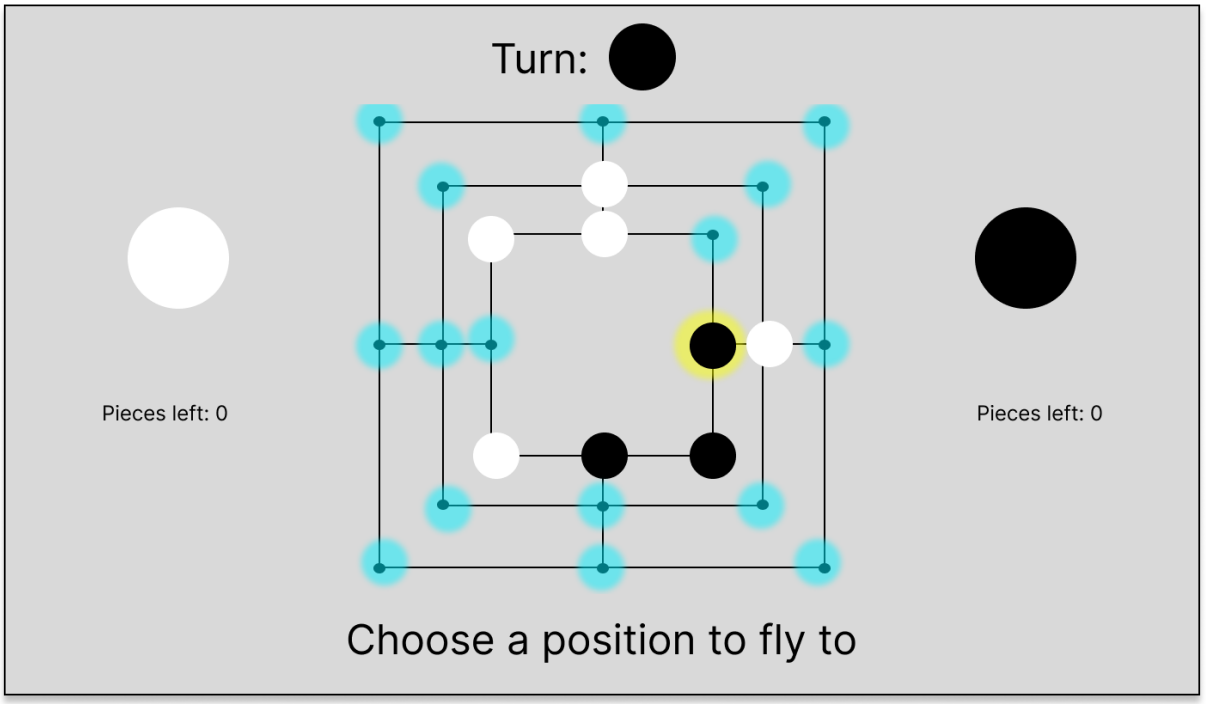


Once a piece is selected, all the positions that can be moved to will glow up.

4.4. Flying

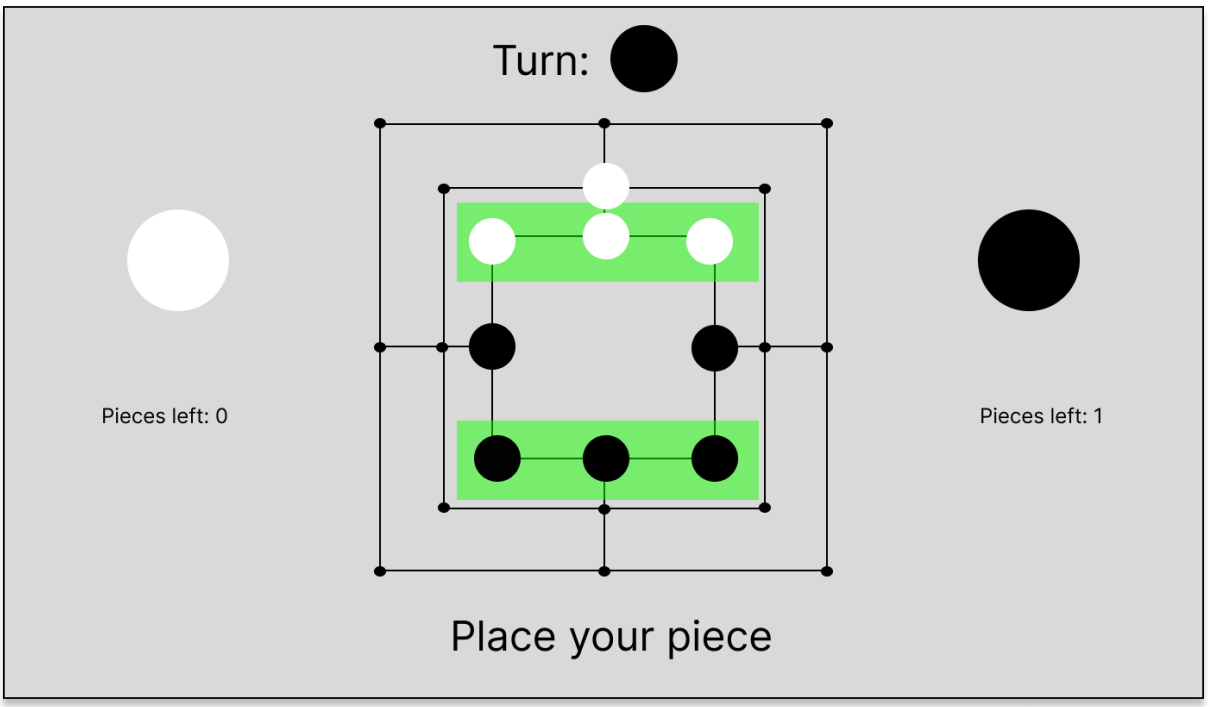


All the pieces that can fly will glow up.



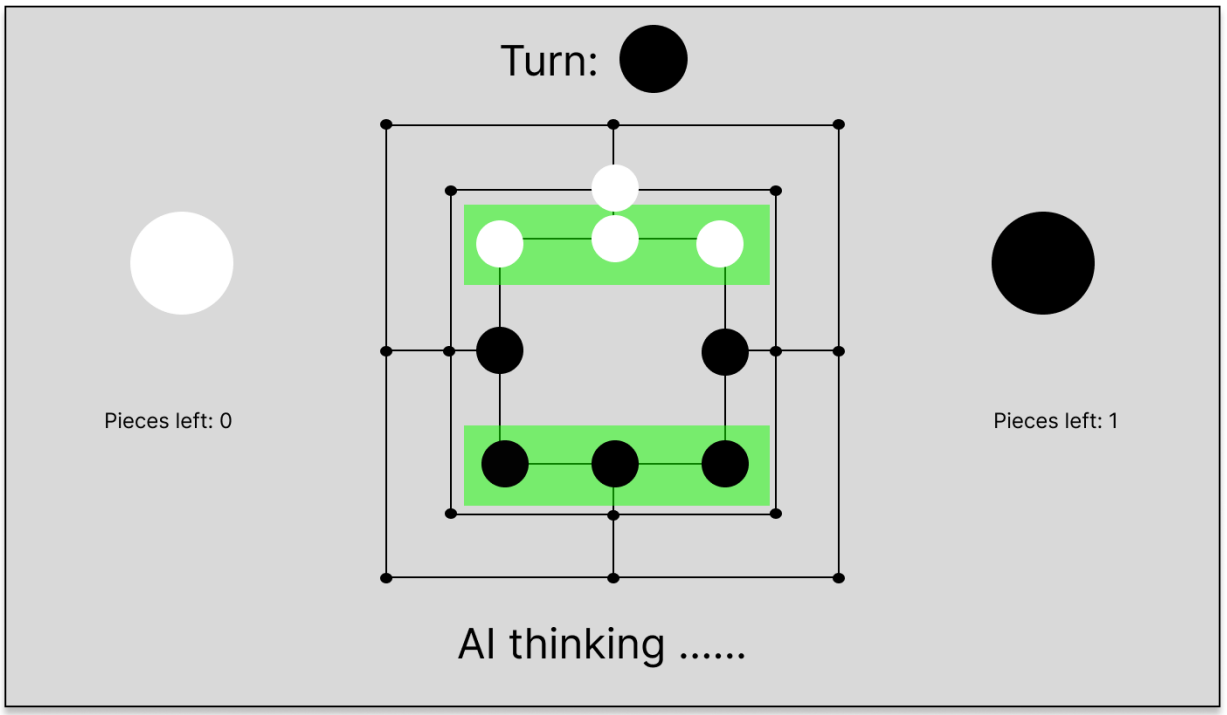
All the applicable positions will glow up.

4.5. Forming a mill



Any mills formed will light as shown above.

4.6. Play against computer



All interactions with the AI will apply, except players will be informed while the AI is calculating its next move.