

Team Membership

Name	Email	Emergency Mobile Contact
Terry Truong	ttru0024@student.monash.edu	0412 773 949
	Technical Strengths: Front End Design, Java, Python Professional Strengths: Teamwork, Dependable Fun Fact: I own 2 cats and I play state league volleyball	
Leon	lmaa0022@student.monash.edu	0432 532 737
	Technical Strengths: Python, Javascript, Java, C Professional Strengths: Logical thinking, problem solving Fun facts: Entered swimming and taekwondo competitions	
Cynthian Thai	ctha0006@student.monash.edu	0490 138 424
	Technical Strengths: Front End Design, Java Object-Oriented Design Professional Strengths: Being concise, Acquiring new knowledge. Fun fact: I have a dog named Louis, I've climbed the Sydney Harbor Bridge	
Yuhan Zhou	yzho0151@student.monash.edu	0422 110 916
	Technical Strengths: Python, Javascript, Application Development Professional Strengths: Teamwork, communication Fun facts: Enjoy playing badminton and video games on a regular basis	

Weekly Meeting Schedule:

We have decided to meet weekly at: Friday 6pm

Work Distribution Model:

Work has been evenly distributed in equal input by each team member for all relevant tasks. Each member has participated in all of our deliverables required and are all expected to withhold this expectation to complete all set tasks.

Our team has also decided to consistently recap what work has been completed each week and what work requires assistance meaning that if one team member is unable to complete a set task they can be assisted by another member of the team. This works to create a cohesive team environment.

Technology Stack Justification:

Language	Pros	Cons	Comments
Java	<ul style="list-style-type: none">Created for object oriented programmingLarge Variety of libraries readily availableEasy to learn, all team members have previous experience in using Java language.	<ul style="list-style-type: none">Not very suitable for 3D modelling gamesLimited performance compared to other languages such as C++ in terms of game development.Limited graphics capabilities	<ul style="list-style-type: none">Very suitable language for this assignment as an object oriented design approach is required.Many of our group members have also had prior experience in using Java, so the language should be comfortable for our team.
Python	<ul style="list-style-type: none">Includes Object Oriented ProgrammingResources are abundant for this language as it has been around for a long timeAlready learnt and used before by all team members <p>Pygame designed for game design</p>	<ul style="list-style-type: none">Speed in comparison to other languages like C++ is lackingLow compatibility with game enginesLimited number of game librariesCannot create own game engine	<ul style="list-style-type: none">Easy to use, not very optimal for serious game development
JavaScript/ HTML	<ul style="list-style-type: none">Very suitable in creating an interface for board games such as 9MM	<ul style="list-style-type: none">-Speed in comparison to language such as C++, Javascript is extremely slow and	<ul style="list-style-type: none">Very suitable for creating usable interface for the user, however may be

	<ul style="list-style-type: none"> • Easy integration to user inputs • All team members have previous experience in using javaScript/HTML 	<p>consumes a lot of memory</p> <ul style="list-style-type: none"> • JavaScript only support single inheritance and this may interfere with efficient of inheritance 	<p>difficult to use to implement object oriented design</p>
--	---	---	---

Front End Framework	Pros	Cons	Comments
Java Console	<ul style="list-style-type: none"> • Easy to use, no additional learning involved • All team members have previous experience in creating a game using the console 	<ul style="list-style-type: none"> • Hard to realistically represent the game, as tokens, player moves have limited representation which may create confusion for the players 	<ul style="list-style-type: none"> • Easiest method of displaying the game • Could take long time coding game representation and connecting display to code logic
JavaFx	<ul style="list-style-type: none"> • Lots of resources available online (e.g. Youtube videos, tutorials) to guide creating a interface • Able to create a game display that represent the game more realistically • Built-in animation system • Media support 	<ul style="list-style-type: none"> • All members do not have previous experience in using JavaFx, therefore hard to predict learnability • Not as mature in terms of getting all the bugs out 	<ul style="list-style-type: none"> • Has a learning curve for all members as no one is familiar with JavaFx but once familiarised would be much better to utilise for designing the game interface
Java Swing	<ul style="list-style-type: none"> • Lots of resources available online (e.g. Youtube videos, tutorials) to guide creating a interface • Able to create a game display that represent the game more realistically • Better Game interface leading to better user 	<ul style="list-style-type: none"> • All members do not have previous experience in using Java Swing, therefore hard to predict learnability • Customising look is more difficult • No built-in data-binding concept 	<ul style="list-style-type: none"> • Has a learning curve for all members as no one is familiar with Java Swing but once familiarised would be much better to utilise for designing the game interface

	experience		
--	------------	--	--

Final justification of language choice:

Java as the project is extremely object oriented in design. Due to this heavy emphasis on object oriented programming that this project demands for, Java's clear strengths heavily outweigh the other design choices. Although python was considered alongside Java due to its UI capabilities over Java using its Pygame inbuilt extension, Java was ultimately chosen due to its suitability for this specific task and also has been used prior by all our team members.

The front-end framework has not been decided yet as the group will have a better understanding of the pros and cons and whether it will fit into our design for 9MM once we start working on the code and get a better understanding of each framework, however we are leaning towards JavaFx as it have more fitting pros and cons compared to Java Swing on paper.

User Stories:

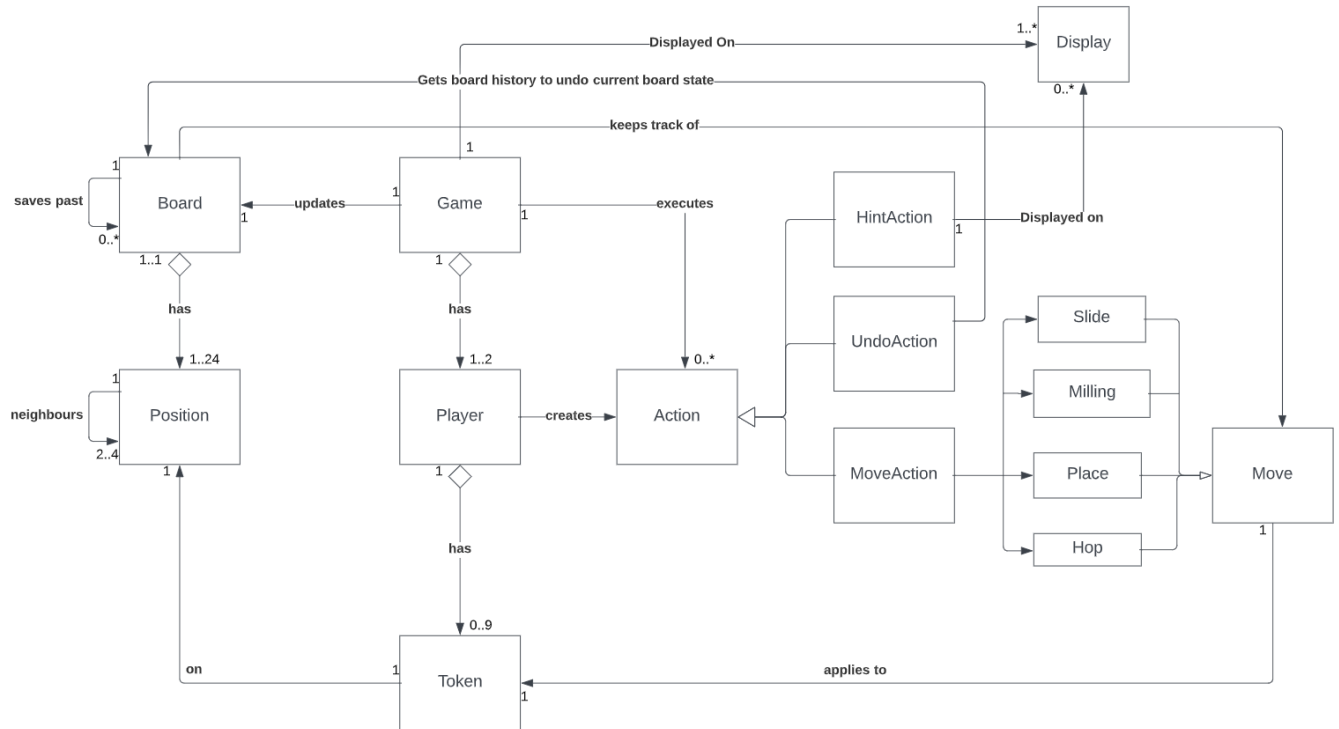
1. As a player, I want to be able to place a piece on the board during my turn so that I can play the game.
2. As a player, I want to be able to see how many pieces I have left to place so that I can plan what I want to do or where I want to put my next piece.
3. As a player, I want to slide my piece on my turn once all my pieces have been placed so that I can create a mill.
4. As a player, I want to be able to create a mill successfully by combining three of my own pieces in a row so that I can remove a piece from my opponent.
5. As a player, I want to be able to choose which one of my opponent's pieces I want to remove so that I can devise a strategy.
6. As a player, I want to be able to see when it is my turn so that I can make my next move.
7. As a player, I want to be able to see the board with the pieces that have been placed so that I can plan my next move and see how I can get a mill.
8. As a player, I want the board to update when I make my move so that the game can continue.
9. As a player, I want to see a victory screen when my opponent runs out of pieces so that I can know if I have won the game.
10. As a player, I want to see my previous moves so that I can analyse the game and see where I could have improved
11. As a player, I want to be notified when I have won the game in order to know for sure that I have won instead of continuing the game
12. As a player, I want my “tokens” to be different colours from my opponents so that I distinguish between the two so don’t move the wrong tokens
13. As a player, I want to be able to easily see the rule book at any time, so I can double check the rules when I’m unsure
14. As a player, I want to be able to click a quit button so that I can quit my game whenever I want.

Advanced Functionality:

1. As a user, I want to be able to save my game when I am quitting so that I can return to the game later.
 2. As a player, I want to be able to undo so that I can remove a mistake that has been made.
 3. As a player, I want to be able to play a tutorial so that I can learn how to play the game.
 4. As a player, I should be able to use hints for valid moves so that I can become more familiar with the rules of the game.
-

Basic Architecture

9 Man Morris Domain Model



9MM Domain Model Rationale:

9 Man Morris being a board game, the primary domain entities will include The Board, Game, Player, Move, Token, Position and Display. The relationship between the game and the board is non-trivial as the board will constantly update the game's states throughout the game.

According to the game rules, there will be two players and each player will need to make a move during their turn, moving their tokens to certain positions. The available moves that a player can make can be generalised as place, hop or slide. However, advanced modifications to the traditional 9MM games are added, allowing the ability to undo a move or ask for a hint. These new functions can be considered as actions that a player can perform during each turn, however, this action cannot be generalised as a move since it does not create a new state of the board. Therefore an Action entity is added to generalise different actions a player may take, whether it be a move action that will create a new state, an undo action which restores a previous state or a hint action that does not affect the state.




The moves a player can make according to the rules also depend on different phases during the game, for example, a player can only slide their tokens once they have placed down all 9 tokens. The phase domain entity is omitted in the domain model as it should be the move entity's responsibility to determine whether a particular move is valid and adheres to single

responsibility principles and create abstraction from other non-related entities. Milling is a special type of move that is triggered by the player making a move action causing another move. The game will check each turn if the player has created a mill, previous mills do not count. If the player has created a mill, the player gets a 2nd Action of Milling, which is removing an opponent's token. This is an alternative to the Player checking if a mill was created or having a Mill Entity which would not follow Open-Closed or Liskov principle. Our solution uses the already existing Move and Move Action that are used for Slide, Place, Hop and extends from this without adding additional code that would need modification later, following Open-Closed principle. It also follows Liskov substitution principle as the Milling is substitutable for its parent, Move class.

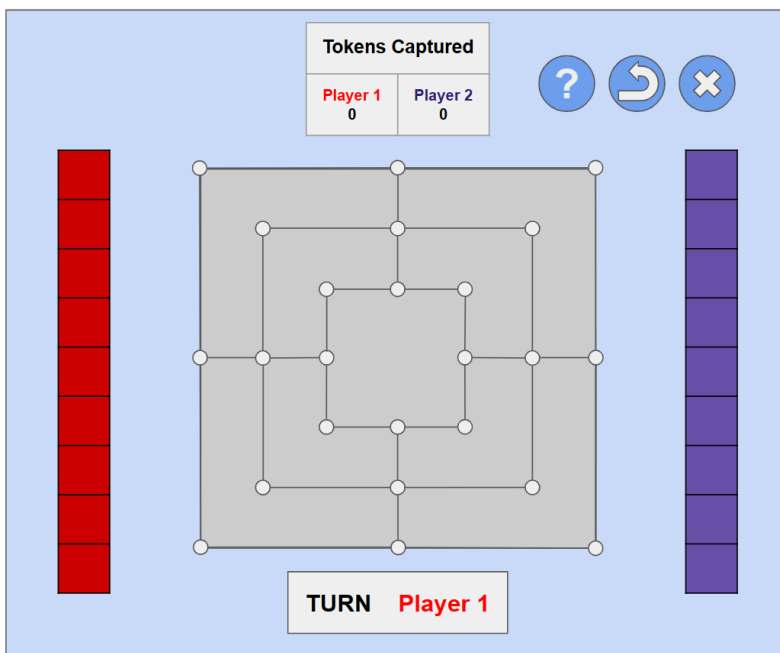
The 9-man Morris board game created will be a computerised version, therefore a display is required to create an interface that the player can see. This display will cover all aspects related to the board, which includes player token position, the board itself and other action buttons such as hint action.

UI Design:

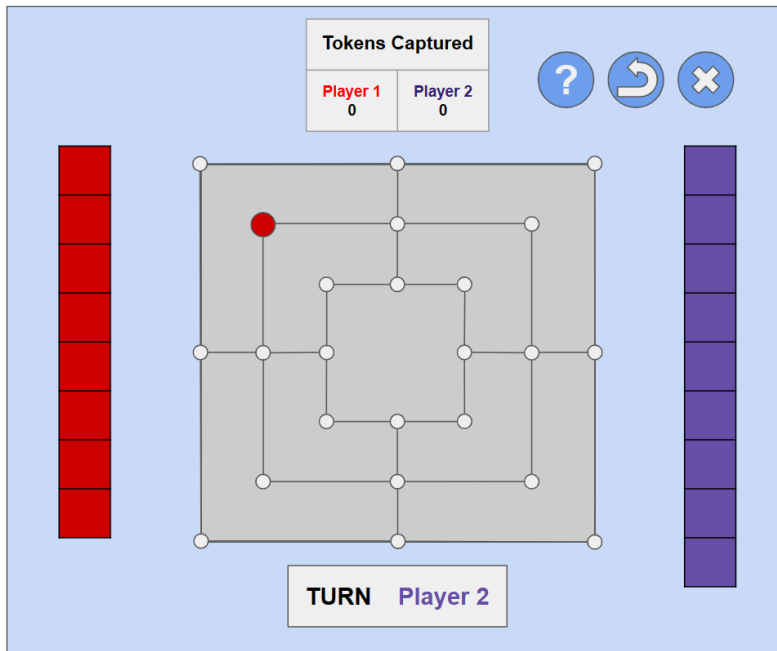
Nine Men's Morris

-  Play new 1v1
-  Load previous save
-  Tutorial

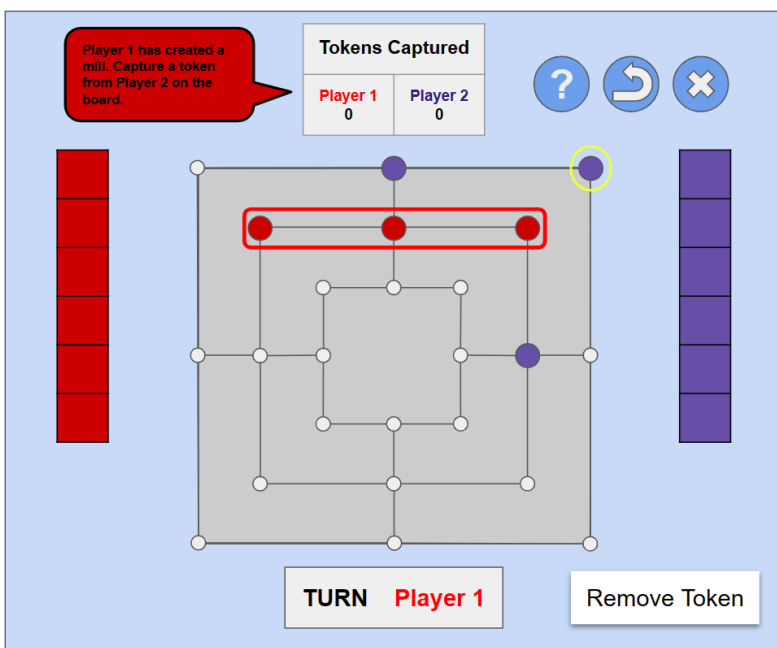
- Main Menu
- Select "Play 1v1"
OR "Load previous save OR "Tutorial"




- Tokens to be placed are on the left (red) and right (purple) of board.
- Player 1 turn indicated below board.

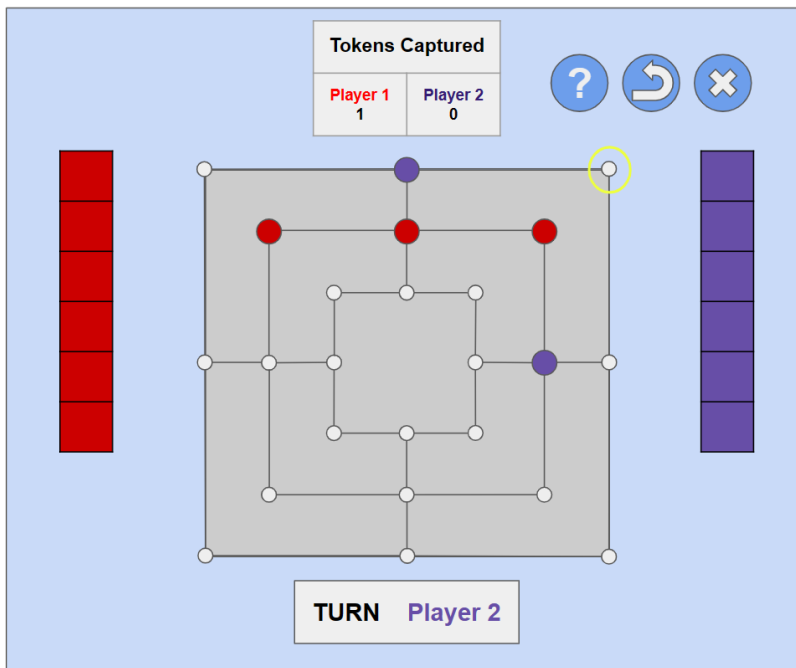


- Player 1 places their token on any position available on the board - indicated by red circle on board.
- Now it is Player 2's turn - indicated at the bottom of the board.

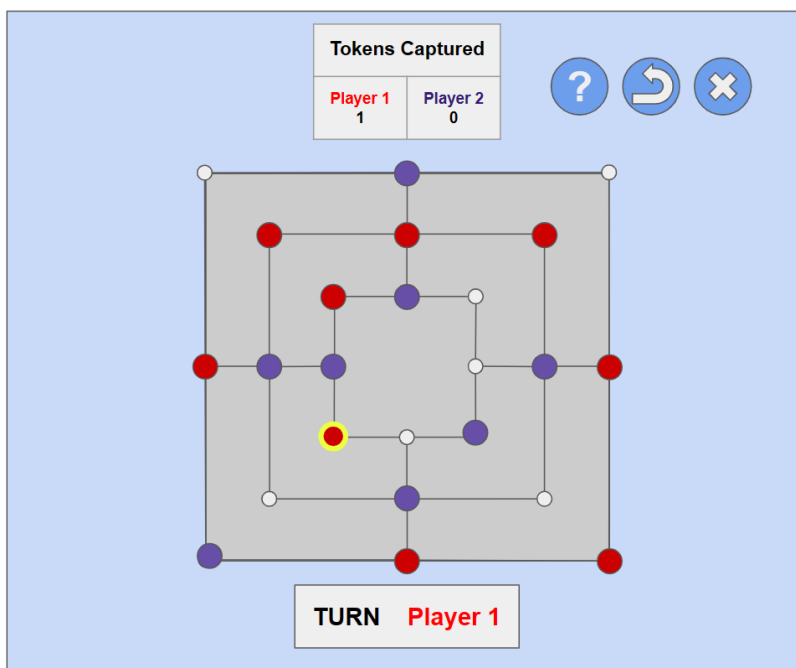


 -Selected Token

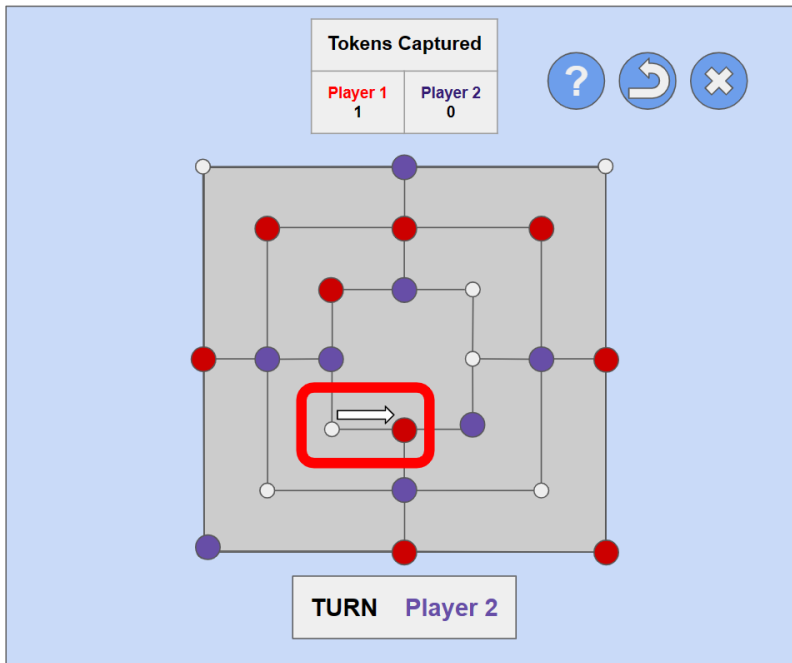
- Player 1 successfully creates a mill (three tokens in a row)
- Player 1 is now able to remove a token from Player 2
- The selected Token will be highlighted in Yellow and confirmation box will appear on the bottom right prompting player to remove Token.



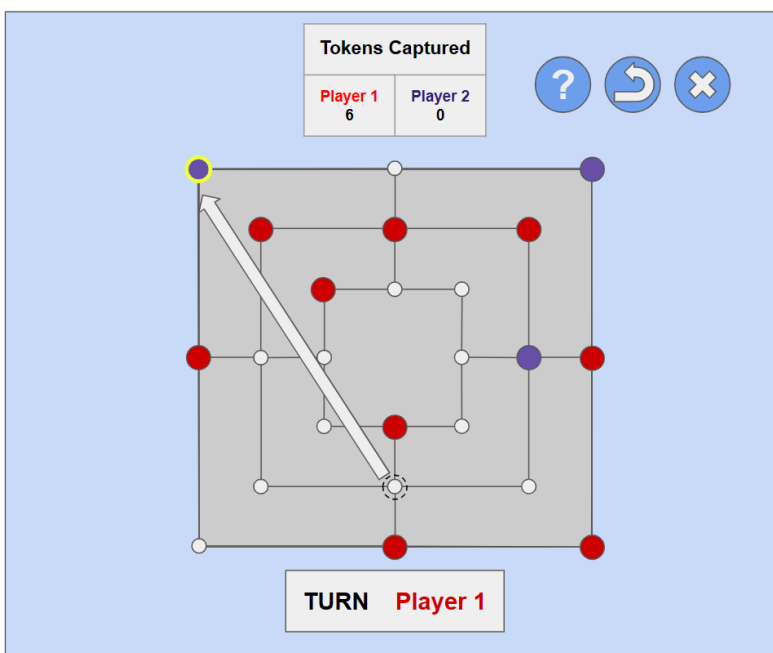
- Player 1 has captured Player 2's token previously in the yellow circle, which has been updated in the score above the board.
- It is now Player 2's turn



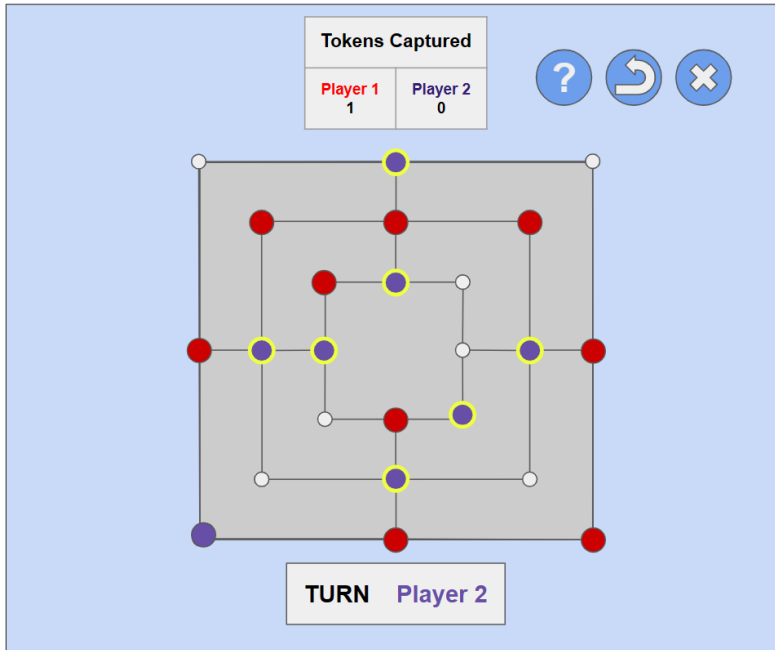
- All tokens from each player have been placed
- Player 1 chooses their token that they want to slide (highlighted in yellow)



- Player 1's token has moved to chosen spot (demonstrated in the red rectangle)
- It's now Player 2's turn.



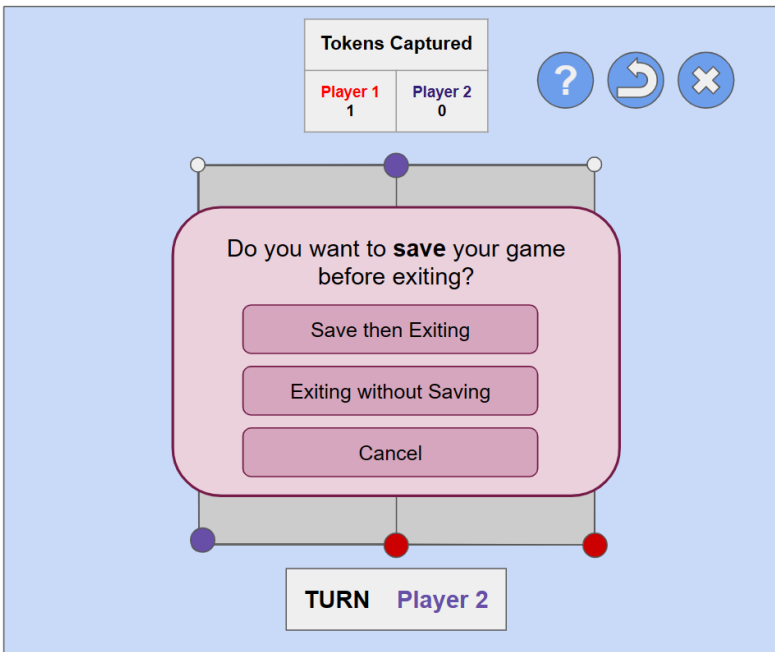
- Player 1 has captured 6 of Player 2's tokens - Player 2 only has 3 tokens remaining on the board so they can now hop.
- Player 2 chooses the token they want to move, highlighted in yellow.
- And can hop anywhere on the board to try create a mill.
- Player 2's token has hopped to their desired position (highlighted in yellow).
- It is now Player 1's turn.



The Player(s) can choose to undo their previous moves



The Player(s) can request a hint on their turn. The tokens that can be moved will be highlighted. Once a token is chosen, their valid positions will be highlighted.



The Player(s) can choose to exit the game using this button

- They will be prompted with this screen and choose their option
- If they choose to exit, they will return to the main menu.