FIT3077 Software Engineering: Architecture and Design Sprint One

Present by Team_15

Content:

Team Information

Team name: Innovatech

Meaning behind this name:

- Driving progress and development through technological innovation.
- Combining the power of innovation and technology to create unique and efficient solutions.

- Fusing creativity and expertise to provide clients with the highest quality products and services.

- Upholding a spirit of innovative technology on the path to excellence.

- Breaking traditions and conventions through unique technology and innovative thinking to create more forward-thinking solutions.

Team photo:

Left:Zijie Zhang Middle: Xinke Wang right:Boyang Ma

Team Membership

Name	E-mail	Strengths	Fun fact
Xinke Wang	xwan0213@student.monash.edu	Proficiency in multiple programming languages, knowledge of IT project management, system administration, database management, and web development.	In the past, I was a huge fan of Blizzard games, but due to some highly questionable actions taken by the company, I have completely severed ties with them.
Zijie Zhang	zzha0340@student.monash.edu	Did engage the fundamental Java and C++, know a bit of OOP rules.	No allergies but picky eaters.
Boyang Ma	bmaa0012@student.monash.edu	Have experience in programming in multiple languages. Have knowledge of IT project management and IT professional knowledge, know how to make a interface well in usability	I love video games and now I try to learn how to use the game engines on websites and how to make a game by using them as a self study.

Team Schedule

Meeting 1: Project Kickoff Meeting
Date: 29/03/2023
Time: 21:00 pm
Attendees: Xinke Wang, Zijie Zhang, Boyang Ma

Agenda:
- Introductions and icebreaker
- Overview of the project goals
- Assignment of tasks and responsibilities
- Complete the team information
- Discuss technology stacks the team needs to use

Meeting 2: Requirements Gathering Meeting
Date: 30/03/2023
Time: 20:00 pm
Attendees: Xinke Wang, Zijie Zhang, Boyang Ma

Agenda:
- Discussion of project requirements
- Identification of user stories
- Determine the software used to draw the domain model and create and share it

Meeting 3: Basic Architecture Review Meeting
Date: 31/03/2023
Time: 19:00 pm
Attendees: Xinke Wang, Zijie Zhang, Boyang Ma

Agenda:
- Discussion of software architecture and design patterns
- Review of design decisions
- Determine the tool used to draw the low-fi prototype

Meeting 4: Basic UI Review Meeting
Date: 01/04/2023
Time: 19:00 pm
Attendees: Xinke Wang, Zijie Zhang, Boyang Ma

Agenda:

- Discuss basic UI design
- Review of low-fi prototype ensure covering all elements of the 9MM game

Meeting 5: Sprint One Review Meeting
Date: 02/04/2023
Time: 19:00 pm
Attendees: Xinke Wang, Zijie Zhang, Boyang Ma

Agenda:

- Review team information and discuss revisions
- Review required technology stacks and discuss alternatives
- Review and revise all user stories to ensure all functionality is covered
- Final review and revise the domain model

Technology Stack and Justification

I prefer to use Java for programming languages because our team members are all proficient in Java and have experience developing programs with it. This will significantly reduce the time required to learn a new language.

To ensure an interface-oriented approach, low coupling and high cohesion, we plan to use design patterns such as Singleton, Factory, and Adapter patterns, among others. This will require us to learn and apply these patterns effectively.

As our objective is to develop a board game that can run independently, we plan to use GUI toolkits such as Java Swing and Java AWT to create the user interface for the desktop application.

If we need to incorporate multiplayer games or online gaming functionality in the future, we may need to use Java Networking API.

User stories

Overall:
As a game player,
I want the color of my opponent's pieces to be different from my own ,
So that I can easily distinguish which pieces belong to me on the chessboard.

As a game player,
I want to see how many pieces remain on my side to place,
So that I can better formulate strategies.

As a game player,
I want to move my token,
So that I can form three tokens in a row.

As a game player,
I want to move my token,
So that I can avoid the opponent putting three tokens in a row.

As a game player,
I want to form three tokens in a row,,
So that I can reduce the number of opponent 's tokens in the gameplay.

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

As a game player,
I want to be able to see a valid gameboard while gaming,
So that I can clearly know where I can place the tokens.

As a game player,
I want to be able to see a valid gameboard while gaming,
So that I can clearly know where I can place the tokens.

As a game player,

I want to place my tokens to the available points on the board,
So that I can participate in the game.

As a game player,
I want to move my pieces on the board to empty adjacent points
So that I can avoid my opponent to get three tokens in a row.

As a game player,
I want to remove the opponent’s pieces on the board
So that I can reduce the number of opponent ‘s tokens.

As a game player,
I want to see the scores in the game,
So that I can know how many rounds I have winned.

As a game player,
I want my pieces to be different from my opponent's,
So that I can clearly identify where the pieces belong.

As a game player,
I want to highlight the pieces I manipulate
So that I can clearly identify which piece is being manipulated by me.

As a game player,
I want to see the number of remaining pieces in placement phase,
So that I can easily know how many tokens of mine are available to move.

As a game player,
I want to see the current game phase is present while gaming,
So that I can understand what actions are available to me.

As a game player,
I want to receive reminders when my pieces form a sequence of three,
So that I can avoid missing any opportunities to make a move and prevent unfair situations caused by oversight.

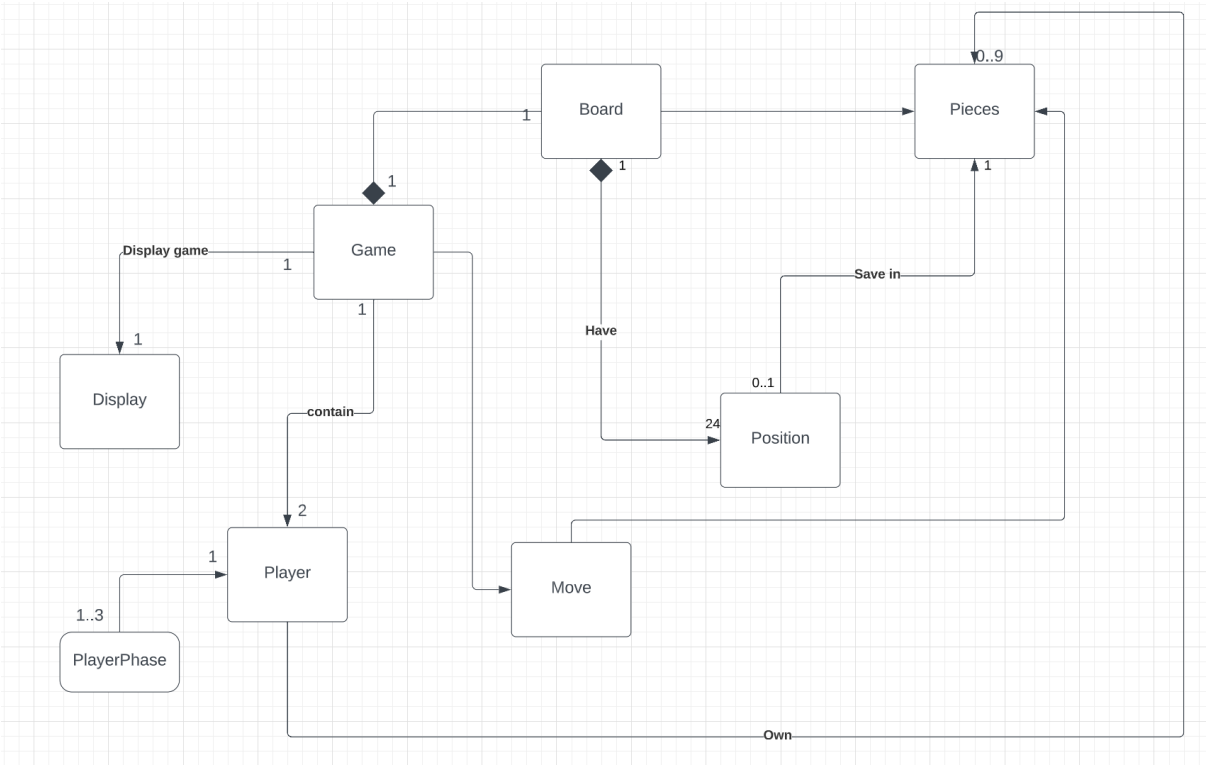
As a new game player,
I want to know the rules of game before gaming,
So that I can get familiar with gameplay faster.

As a new player,
I want to have a trial mode for me to play ,
So that I can know how to play the game.

Basic architecture With design rationale

(Nine Man Morris will represent with 9MM)

Domain Model



Explanation for domain Entities:

- Game Class: This class represents the entire game, including attributes for the current game phase and the current turn, as well as methods for initiating and concluding the game. When the player starts the game, this class will initialize the attribute for the player to create a new game.
- Player Class: This class represents a player in the game and contains attributes for the player's name, score, and color (black or white).
- Board Class: This class represents the game board, including attributes for the board layout and current game phase, as well as methods for placing, moving, and removing pieces on the board.
- Piece Class: This class represents an individual piece on the board, including attributes for the piece's color, position, and whether or not it has been placed on the board. It also contains methods for moving the piece.
- Move Class: This class represents a player's move and contains attributes for the starting and ending positions of a piece. When the player moves the pieces, the system will call the method of move to move the piece on the board.
- Position Class: This class represents the position of the board, this class will restore the position on the board and the position of the pieces which are placed on the board. When the player moves and places the pieces, only the empty spaces can allow the player to move.

Explanation for Associations (design rationale included as well)

- The Game class has a one-to-two association with the Player class, as a game can have 2 players but each player can only participate in one game. (The Game class can also create players.)

  In the 9MM , there are always two players playing against each other. This means that there must be two Player objects for the game to proceed. By creating a one-to-two association between the Game class and the Player class, we were able to accurately model the realistic constraints of the game.

  By creating a one-to-two association between the Game class and the Player class, we ensure that the Game class has access to exactly two Player objects that can be moved around the game board. This

association allows the Game class to manage the game state in a controlled way and ensures that the game can be played according to the rules of 9MM.

- The Player class has a one-to-many association with the Piece class, as a player can have multiple pieces on the board but each piece belongs to only one player.

  Encapsulation and data hiding: Our Player class is associated with a collection of Piece objects to encapsulate the player's actions and reduce the exposure of pieces, improving the overall design quality of the code by reducing the complexity of interactions between different objects.

  Reusability: In games similar to 9MM, they tend to increase the number of pieces, the size of the board, etc. If we were to create a variation of the game of nine chess that allowed more than two players, we could easily modify the code to create a one-to-many association between the Player and Piece classes.

- The Board class has a composition association with the Piece class, as the board is made up of multiple pieces and the existence of the pieces is dependent on the board.

  Simplify: With combinations, pieces can be created along with the board. This reduces the complexity of the code and makes it easier to maintain and understand.

  Those remaining have the same speciality as we summary above.

- The Piece class has an association with the Board class, as each piece is placed on the board and its movement is limited by the board layout.
- The Move class has an association with the Piece class, as it involves moving a piece from one position to another on the board.
- The Position class has a composition association with the Board class. The Board is made up of all the positions on the board. The existence of the position is dependent on the board. If the Board class no longer exists, the Position class will also no longer exist.
- The Board class has a composition association with the Game class. The Board class is part of the Game class and the existence of the

Board class requires the existence of the Game class. If the Game class disappears, the Board class will also disappear.

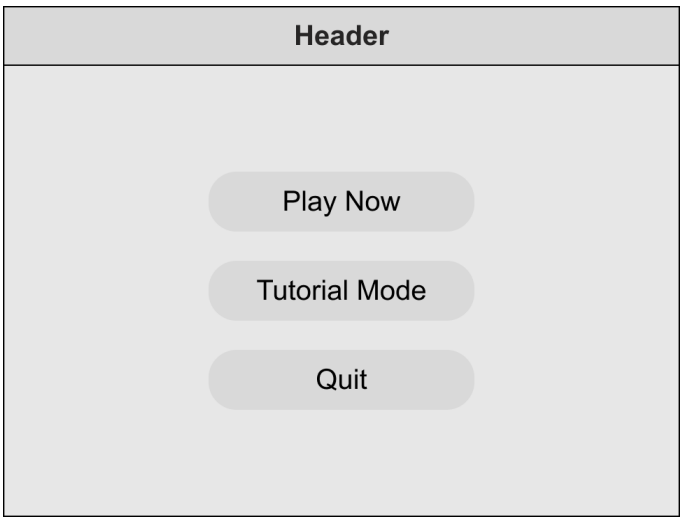
Rationales(Summary)

The domain model employs association and composition relationships to illustrate the connections between different classes. For instance, the Board class has a composition relationship with the Position class, indicating that a Board object contains Position objects. This relationship reinforces the association between the Board and Position classes and ensures that the Position objects are created and deleted along with the Board object.

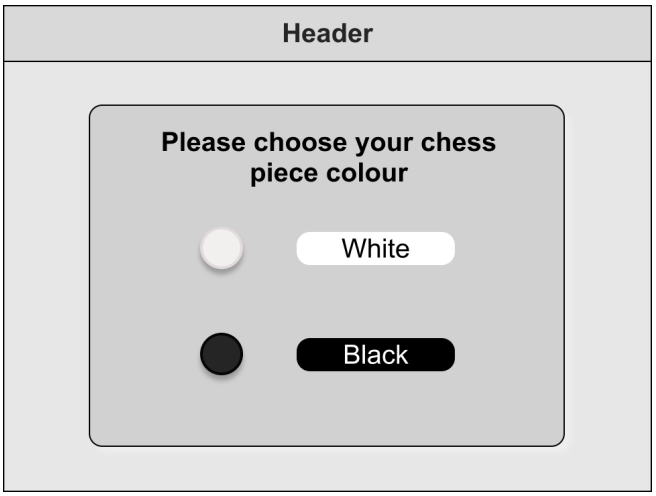
Moreover, the domain model utilizes multiplicity to specify the number of instances of each class that can exist in the game. For example, the Board class has a multiplicity of 1, which restricts the game to have only one instance of the Board class. This approach enforces the singleton pattern for the Board class.

Basic UI design

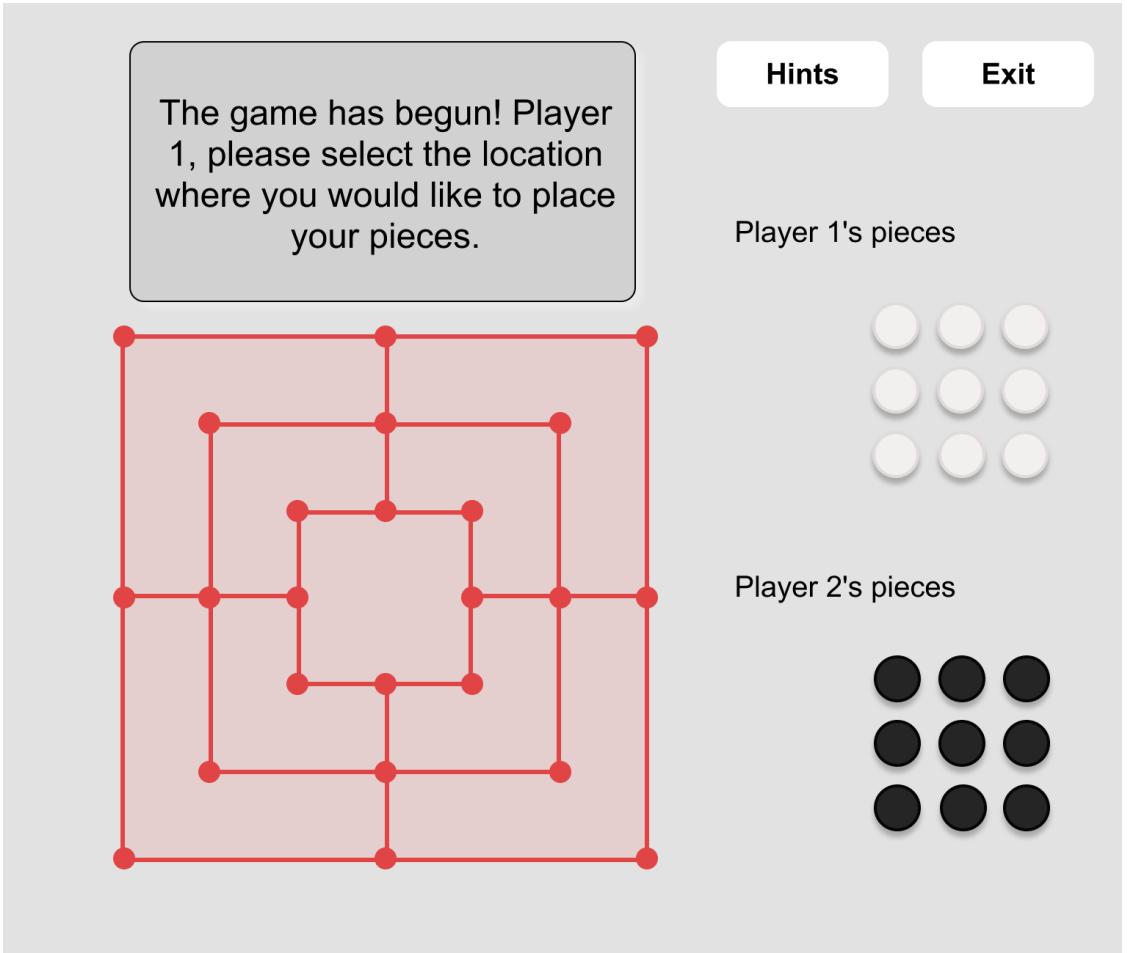
Start interface



Choose color interface



Start placing pieces interface



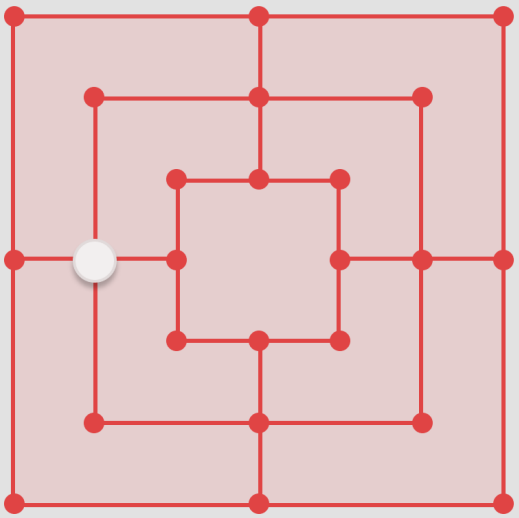
Player 2, please select the location where you would like to place your pieces.

Hints

Exit

Player 1's pieces

Player 2's pieces



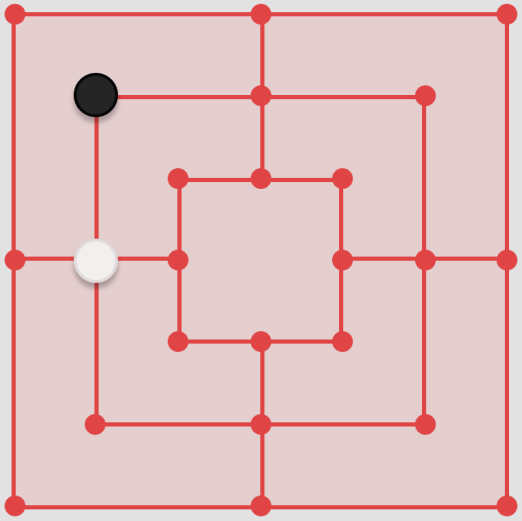
Player 2, please select the location where you would like to place your pieces.

Hints

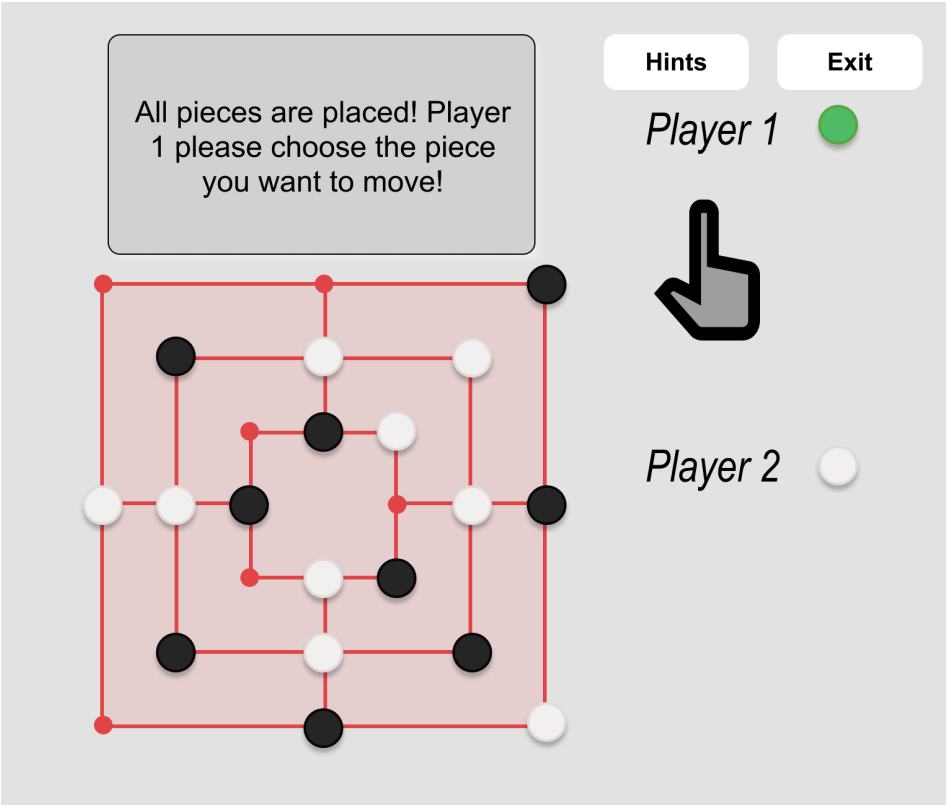
Exit

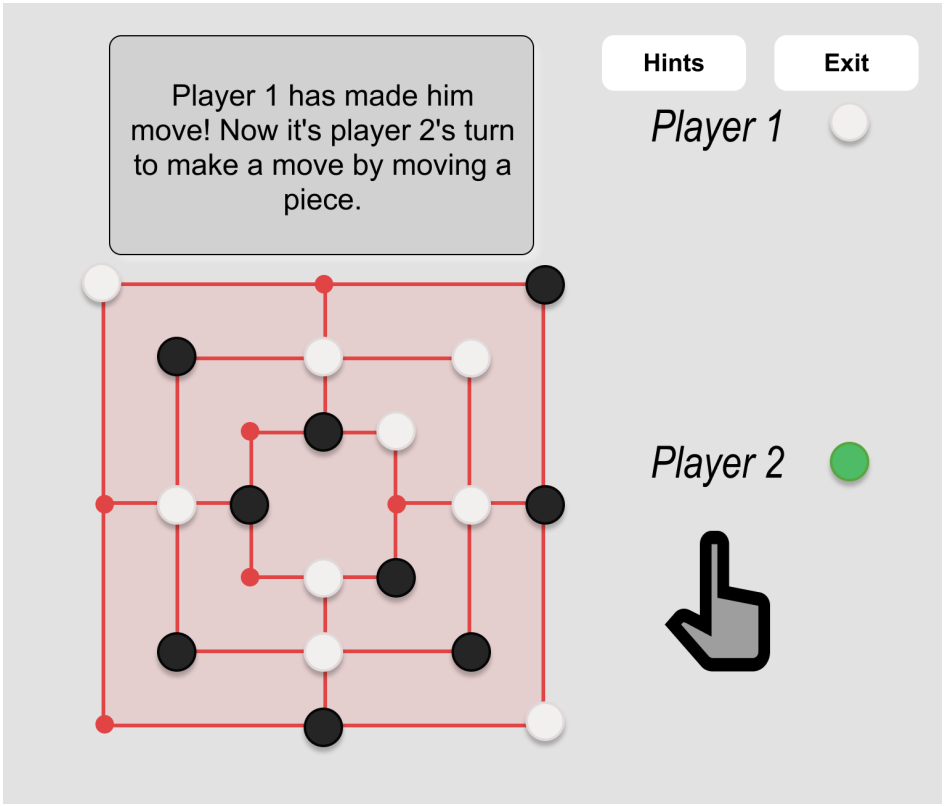
Player 1's pieces

Player 2's pieces

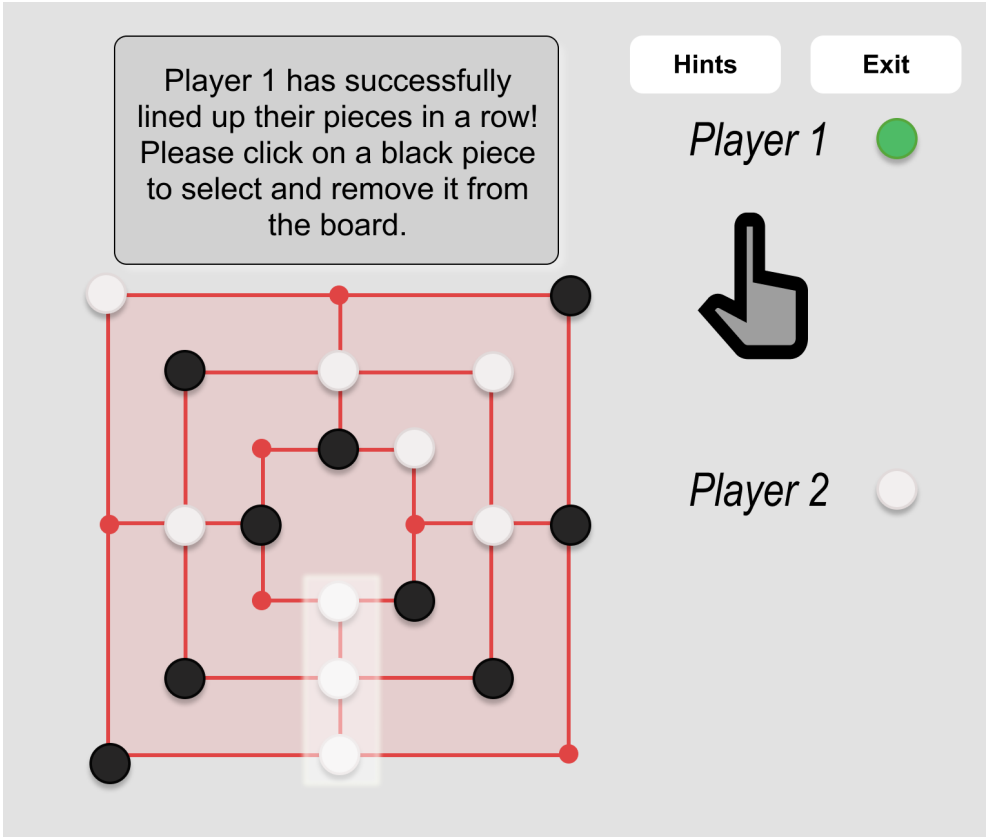


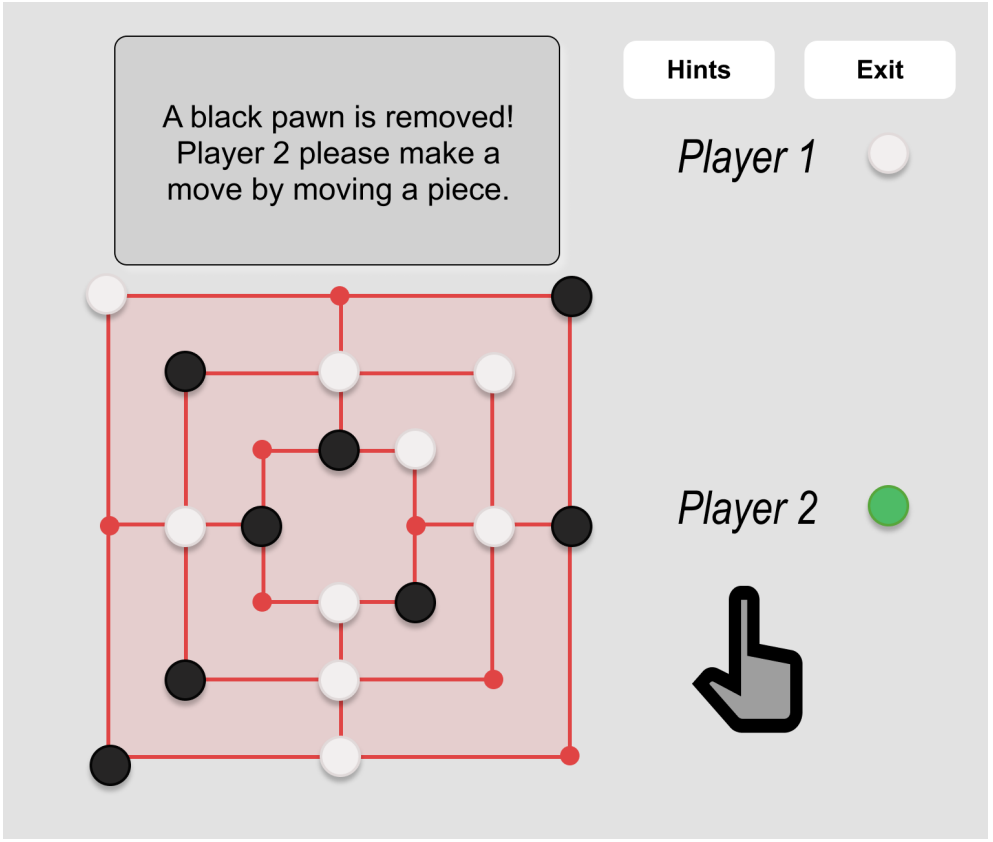
Moving pieces interface



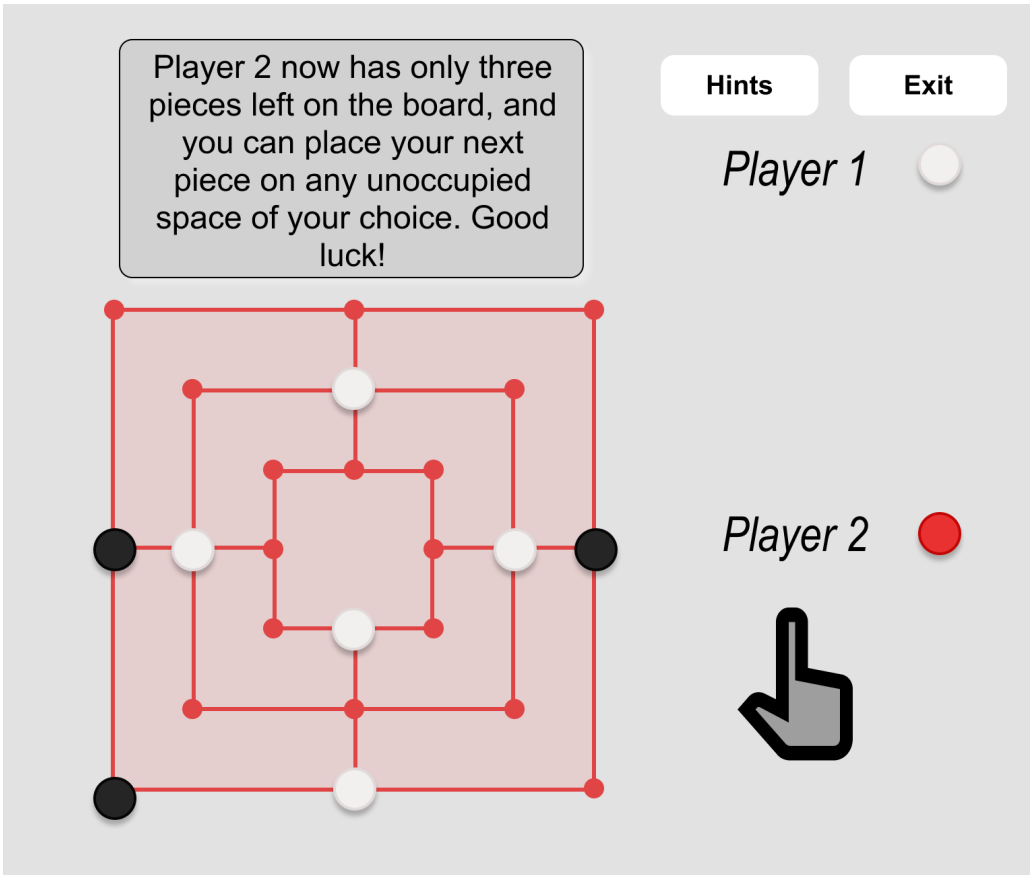

Player 1 has made him move! Now it's player 2's turn to make a move by moving a piece.
Hints
Exit
Player 1
Player 2

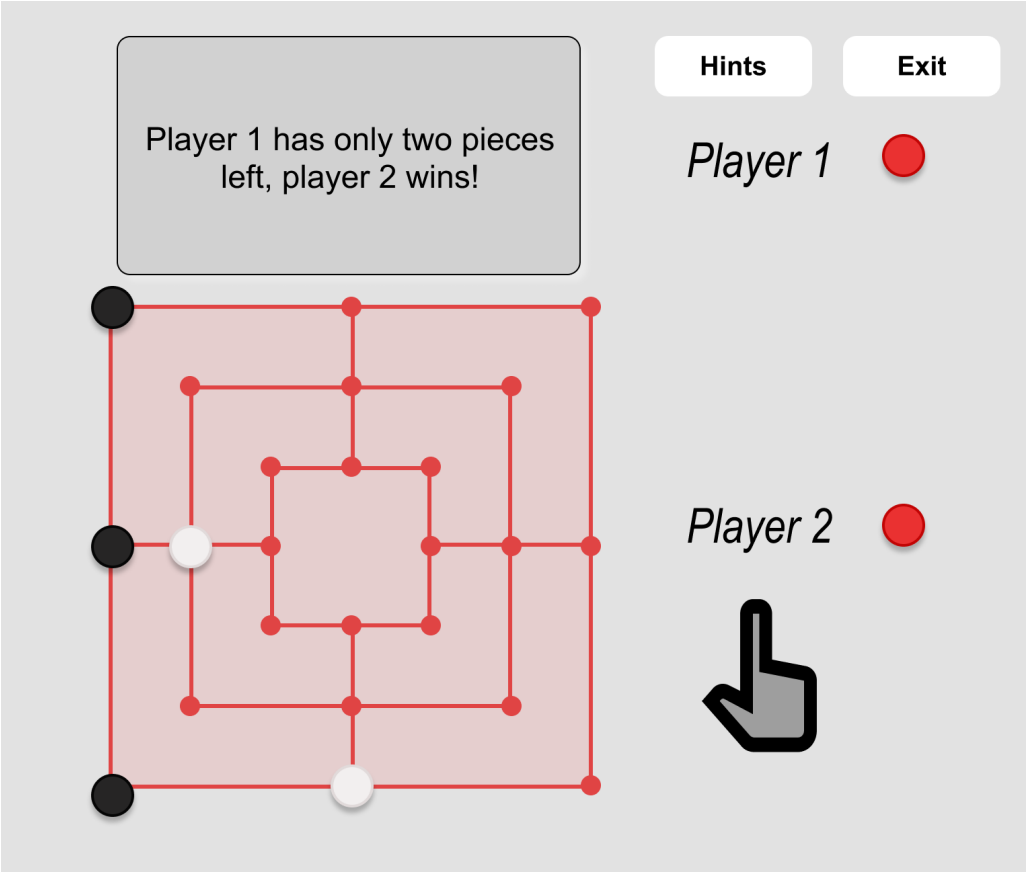
Connect three pieces and remove the pieces



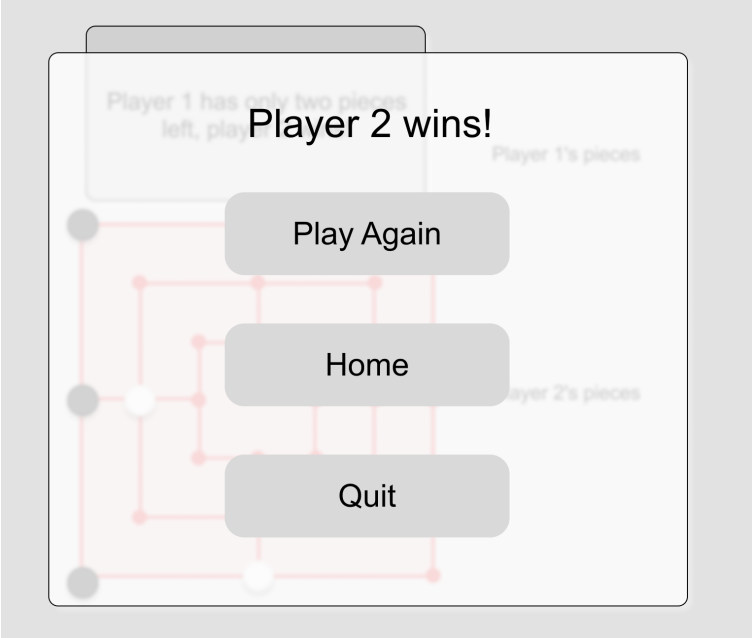


The interface after only three chess pieces are left

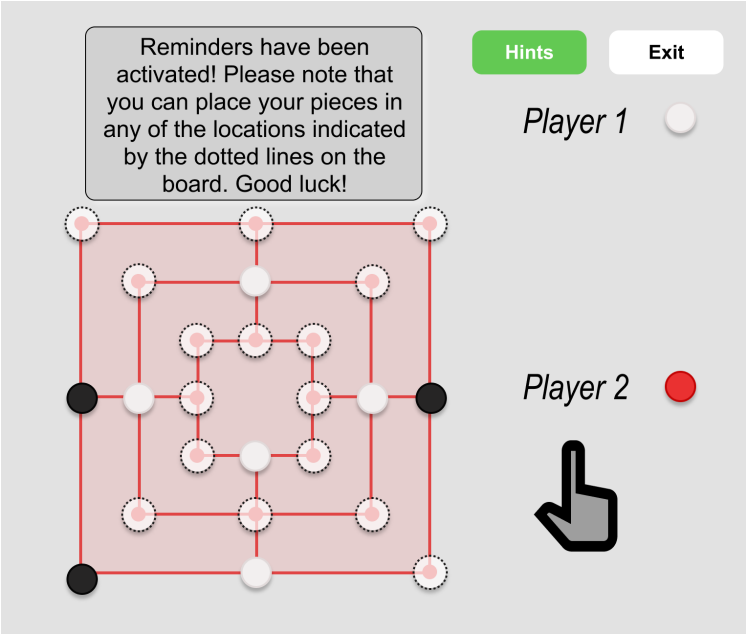



Player 1 has only two pieces left, player 2 wins!
Hints
Exit
Player 1
Player 2

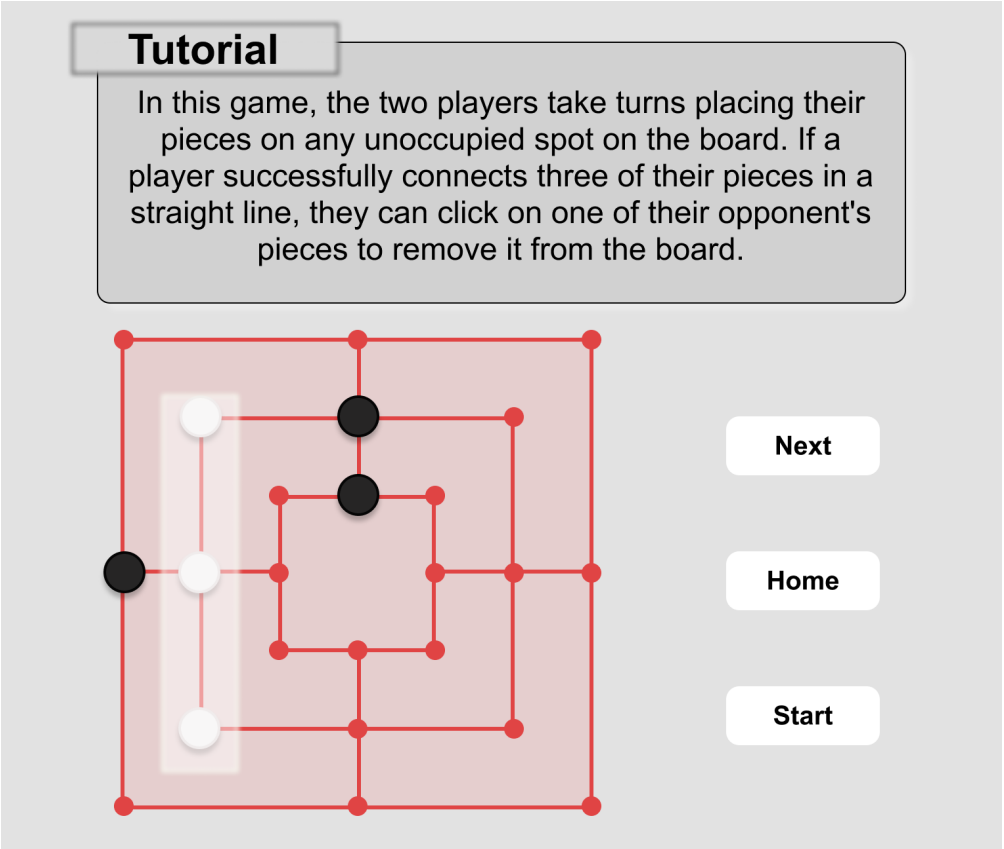
Game over interface

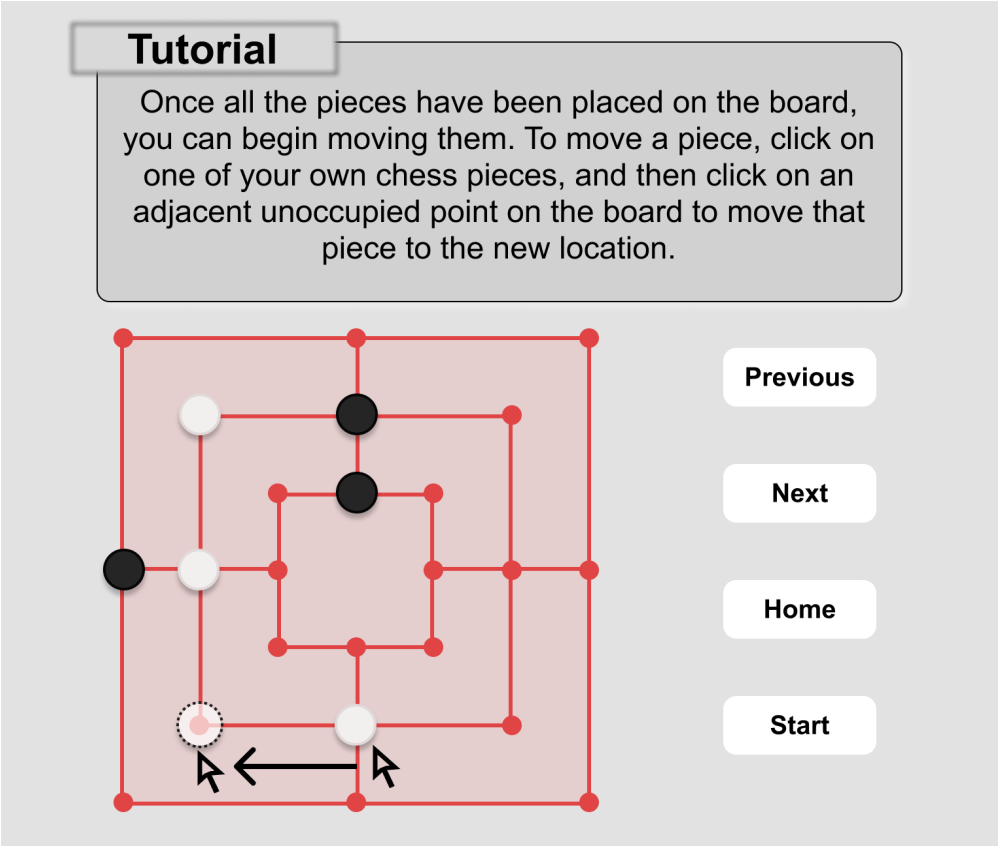


Hints interface



Tutorial mode





Tutorial

When it is your turn and you are in the move phase, there will be a green light behind the player's name. The light will not come on when it is not your turn. When you have only three pieces left to move freely, the color of the light is red.

Player 1

Player 1

Player 2

Previous

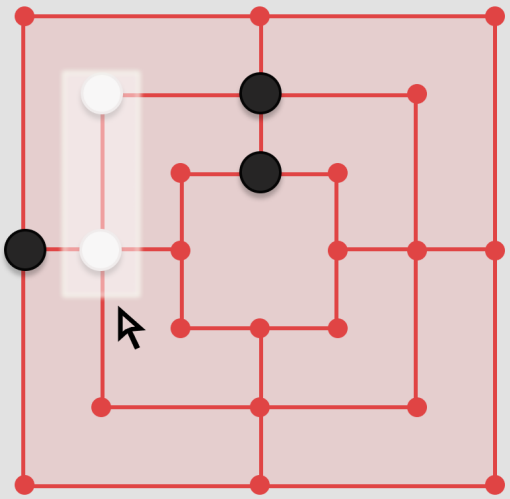
Next

Home

Start

Tutorial

In this game, the objective is to leave your opponent with less than three pieces on the board. If you successfully remove enough of your opponent's pieces such that they have less than three remaining, then you win the game. Good luck and have fun!



Previous

Home

Start