Sprint 3 Deliverables

Design Rationales

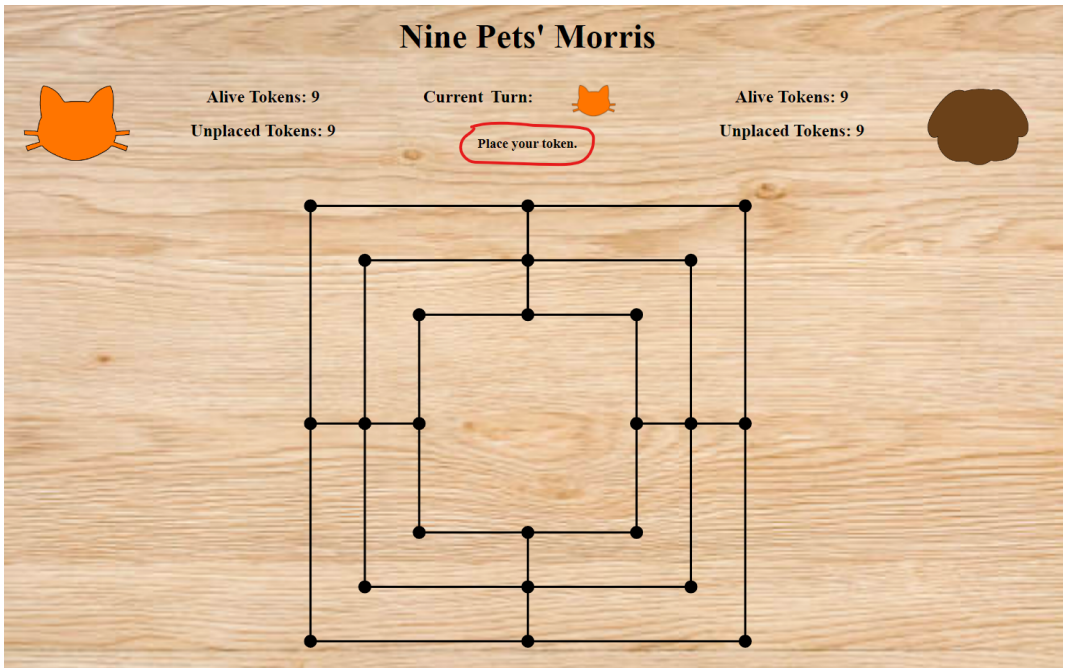
Quality Attributes

1. Usability

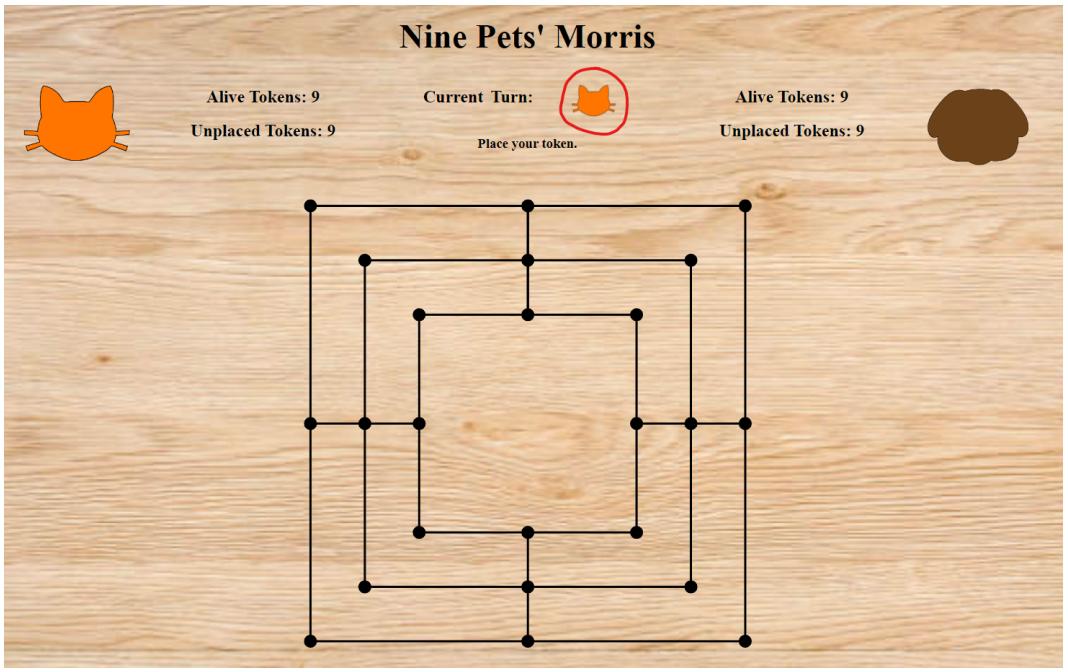
In our implementation of the Nine Men's Morris game, usability is a very important non-functional requirement as there are multiple phases in each round (picking a token up, placing it down and potentially removing an opponent's token), and players take turns doing these actions. This can result in the required action for each phase in each round getting confusing.

In addition, with our advanced requirement, undoing moves, it would make it more difficult to keep track of whose turn it is and what phase the round is on. Therefore, usability is a key non-functional requirement that we focused on.

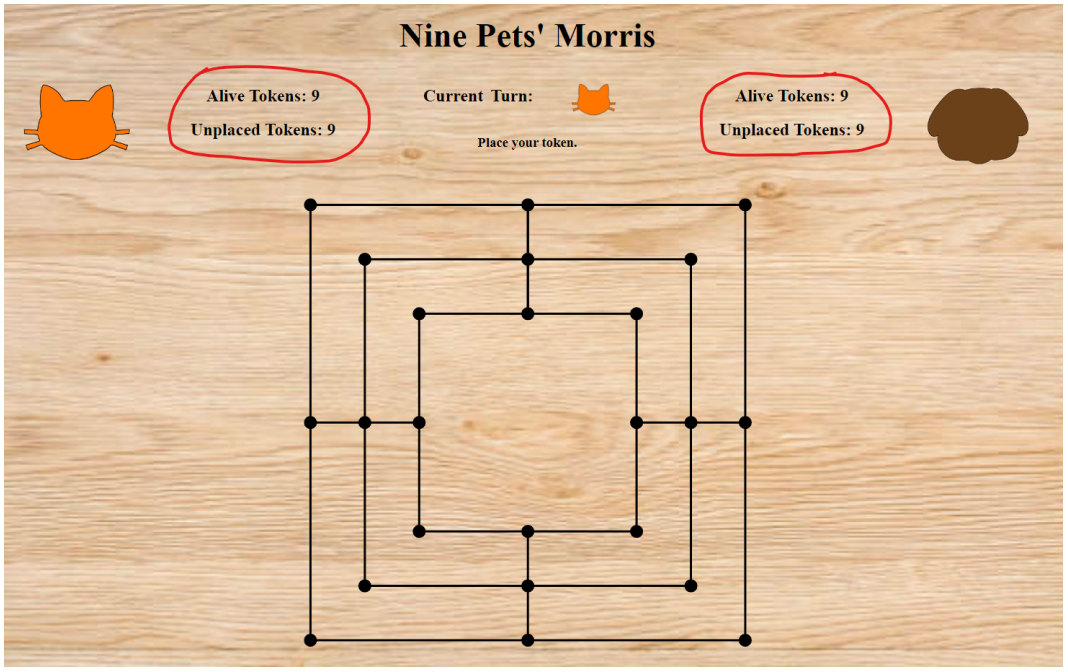
To address the first problem of confusing phases, we added a message above the board to indicate what the player should do for each phase during each round. For each of the phases - picking up, placing down and removing tokens, the messages - "Pick up one of your tokens.", "Place your token." and "Remove an opponent's token!" will appear respectively.



To address the second problem of unclear turns, we added a current turn display, informing the players who should be playing during the current turn. In addition, we used unique and easily identifiable tokens in the shapes of cat and dog to allow the players to easily identify who they are playing as.



Since we do not want our players to have to count the number of tokens on the board or remember how many tokens are left/were removed, we added two counters for each team, showing the number of alive tokens and the number of unplaced tokens. This will allow the players to more easily track the game state. Besides, these counters are displayed with the respective team's token image to more easily highlight who those counters belong to.



2. Correctness

Correctness is another critical non-functional requirement for the Nine Men's Morris game because it defines the degree to which the game satisfies its specifications, rules, and behaviours expected by the players. A 9MM game that is not correct will not be able to provide an enjoyable or fair gaming experience, which can result in player frustration and dissatisfaction.

In implementing the 9MM game, correctness is particularly relevant because it involves the game's rules, player input, and game outcomes. The game should accurately reflect the rules of 9MM, ensuring that players can play the game as they would in a physical setting. For example, the game should allow only the player of the current turn to place or move one of their tokens to a valid position and remove an opponent's token when applicable.

The game should also display the correct game information, such as the number of tokens left for each player and whose turn it is to play. In addition, the game should check if any win conditions are met at the end of each round and display the outcome correctly, ensuring that the winner is verified and determined based on the game's rules.

Furthermore, for our advanced requirement, the game should also allow players to undo each of the previous moves one by one until there is no move left. When implementing this feature, it is essential to ensure the correctness of the game, such that after each undo action, the integrity of the game state is maintained.

Our team ensures the correctness of the 9MM game by implementing methods that validate each move against the game's rules. For instance, the placeToken method in PlaceTokenAction class is invoked every time a player tries to place a token during their turn. In this placeToken method, there are multiple logic checks, such as checking whether the new position to place the token is unoccupied and valid, to ensure that the player is playing by the rules.

```
/**
 * Places a token on the board at position index.
 * @returns true if the token placement was successful, false otherwise.
 */
private placeToken(): boolean {
    let position = this.getBoard().getPositions()[this.positionIndex];
    let pickUpPosition = undefined;
    if (this.pickUpPositionIndex != undefined) {
        pickUpPosition = this.getBoard().getPositions()[this.pickUpPositionIndex];
    }
    let currentTeam = this.getBoard().getPlayingTeam();
    let currentPlayer = currentTeam.getPlayer();

    // if this place token action is a part of moving a token
    if (pickUpPosition) {
        // place the picked up token if the specified position is unoccupied and is a neighbour or if the playing team has 3 or less alive tokens left
        if (position.getPlayer() == undefined && (pickUpPosition.isNeighbour(this.positionIndex) || currentTeam.getNumAliveTokens() <= 3)) {
            position.placeToken(currentPlayer);
            currentTeam.placeToken();
            return true;
        }
    } else {
        // place the new token if the specified position is unoccupied
        if (position.getPlayer() == undefined) {
            position.placeToken(currentPlayer);
            currentTeam.placeToken();
            return true;
        }
    }

    return false;
}
```

Besides, we also tested our 9MM game against different input scenarios and edge cases to ensure that the game's correctness is maintained. For example, we played and tested the game to ensure that players cannot move any of their tokens before they have placed all their tokens, or that they cannot “fly” their tokens until they have only 3 alive tokens remaining, while checking that all game state information displayed in the UI is accurate during each round.

3. Maintainability

Another non-functional requirement we focused on is maintainability. It represents the degree of effectiveness and efficiency with which the product can be modified to improve or adapt to changes in requirements. It is important to the Nine Men’s Morris game because it allows us to change or add features to the game easily, such as when we are implementing our advanced requirement.

In order to implement a maintainable application, we aim to write modular code with adequate documentation.

Having modular code means that we can change one component of the system without having significant impacts on other components. In our implementation, all code is written in logical and modular components with the use of classes, each containing only the relevant methods. For example, we have a Team class representing a team in the game. In the Team class, each method only has a single responsibility. For instance, the placeToken method is only used for decreasing the team’s unplaced tokens count after placing a token, whereas the removeToken method is only used for decrementing the team’s alive tokens count after a token is removed from the game. This ensures that when modifying one component, such as adding a new feature or fixing a bug in the placeToken method or removeToken method, it will have minimal impact on other components. This makes it easier to improve and correct the game.

```
/**
 * Decrements the number of unplaced tokens by one.
 */
placeToken(): void {
    if (this.numUnplacedTokens > 0) {
        this.numUnplacedTokens--;
    }
}
```

```
/**
 * Decrements the number of alive tokens by one.
 */
removeToken(): void {
    this.numAliveTokens--;
}
```

In addition, all code is well-documented with docstrings and in-line comments, in accordance with the Typescript coding standard and style guide. Such documentation allows any programmers to easily understand the structure of the code, even if they

are completely new to the project, and makes the project more maintainable in the long term.

Human Values

1. Creativity

We have considered the human value of creativity in our design of the Nine Men's Morris game because we believe that incorporating creativity into the game can enhance the user experience and provide additional engagement and enjoyment.

To bring our creativity to the game, we have adopted a "pet store" theme for our game, which is also the name of our team. Instead of the default black and white tokens, we used tokens in the shapes of cat and dog, making the game visually appealing and engaging. This design choice adds a unique and creative element to the game that helps differentiate it from other similar games, making it more memorable to players.

Moreover, the use of cat and dog tokens can also make the game more relatable to players who have pets or like animals, which can enhance their overall enjoyment of the game. By tapping into this human emotion, we are not only creating a more engaging and enjoyable experience, but also establishing an emotional connection with our players.

2. Helpful

Another human value that we have considered is helpful. We believe it is relevant and important to our game because it can make the game easier to understand and play, and hence more accessible and enjoyable for players, especially those who are unfamiliar with the game.

One example of how we have incorporated the value of helpful is by adding a section above the game board that displays important information for the players. This information includes the number of alive and unplaced tokens for each team, the current player of each round, and the actions the current player should take. By providing players with this information, it reduces confusion and frustration that can arise from not understanding the game's rules or the current game state, making the game more accessible to players who are new to the game.

Moreover, we have chosen to differentiate each team's tokens not just by colour, but also by shape (cat and dog). This helps players to easily identify which tokens are theirs, which can be especially helpful for players with colour blindness.

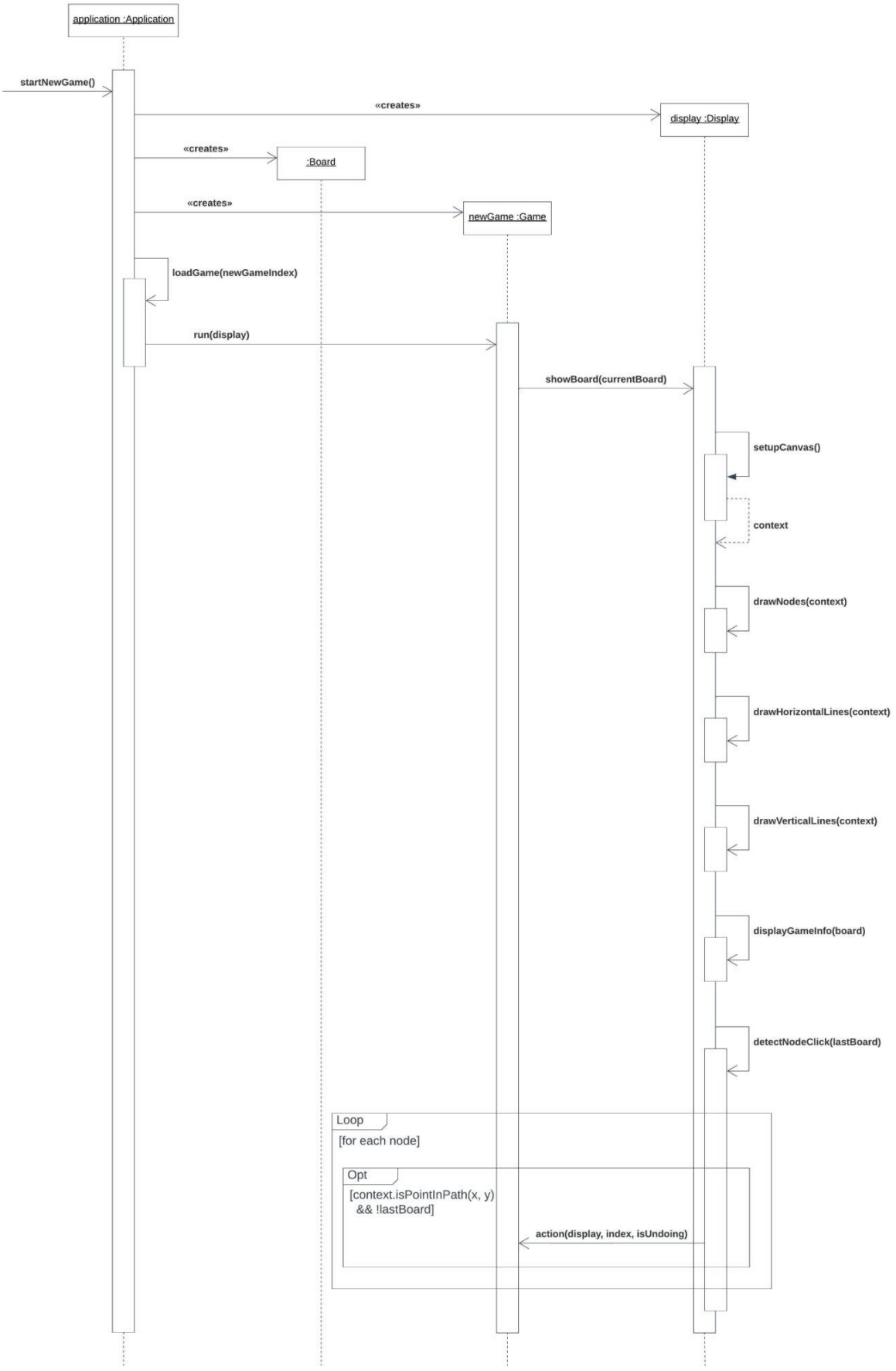
These helpful features can enhance the user experience and make the game more enjoyable for a wider range of players, by reducing frustration and confusion that can arise from not understanding how to play the game correctly or not being able to easily tell which tokens belong to which team.

Demonstration Video Link

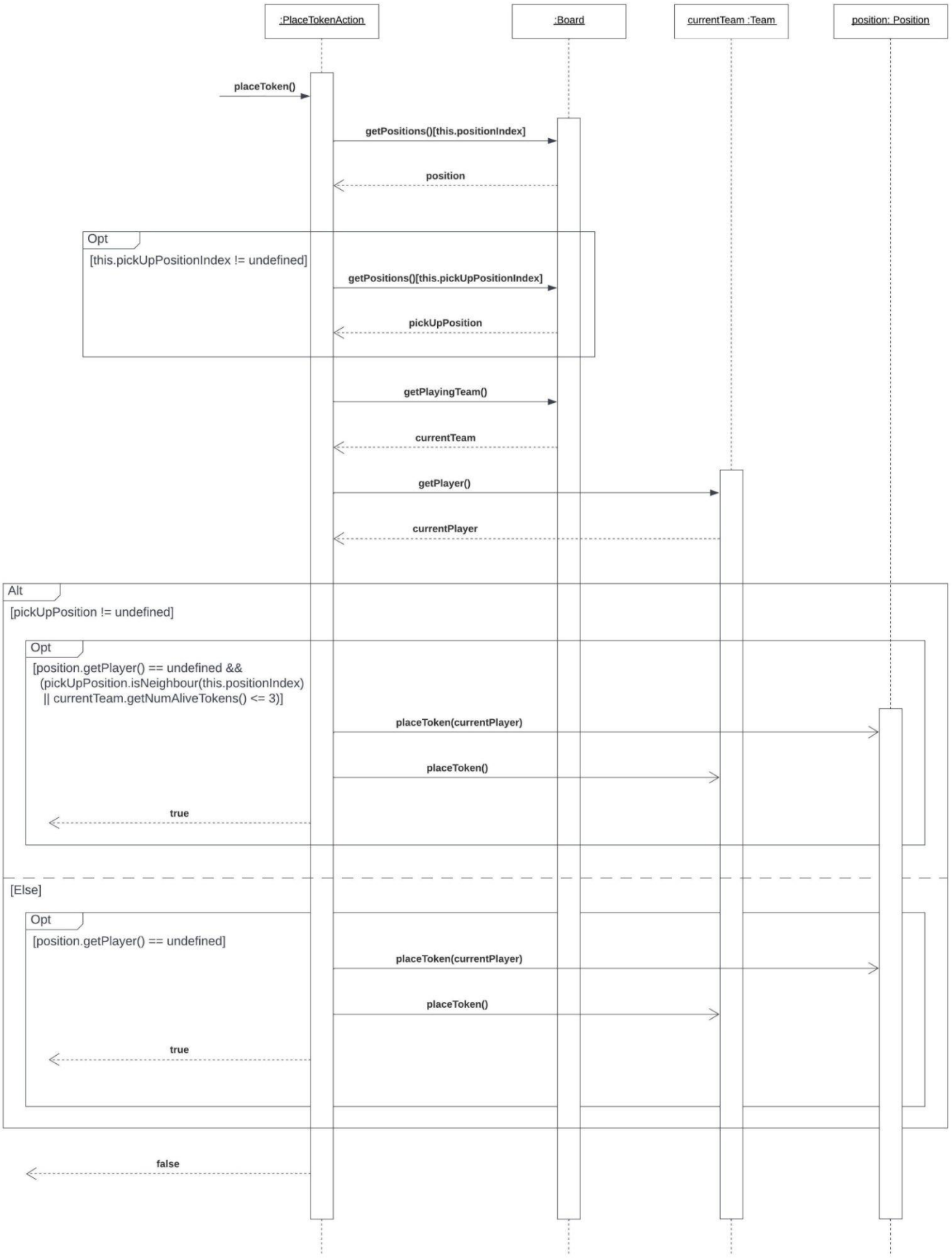
Please click the following link to watch the demonstration video for our implementation of the Nine Men's Morris game: https://youtu.be/HRHcCw7QSfo

Sequence Diagrams

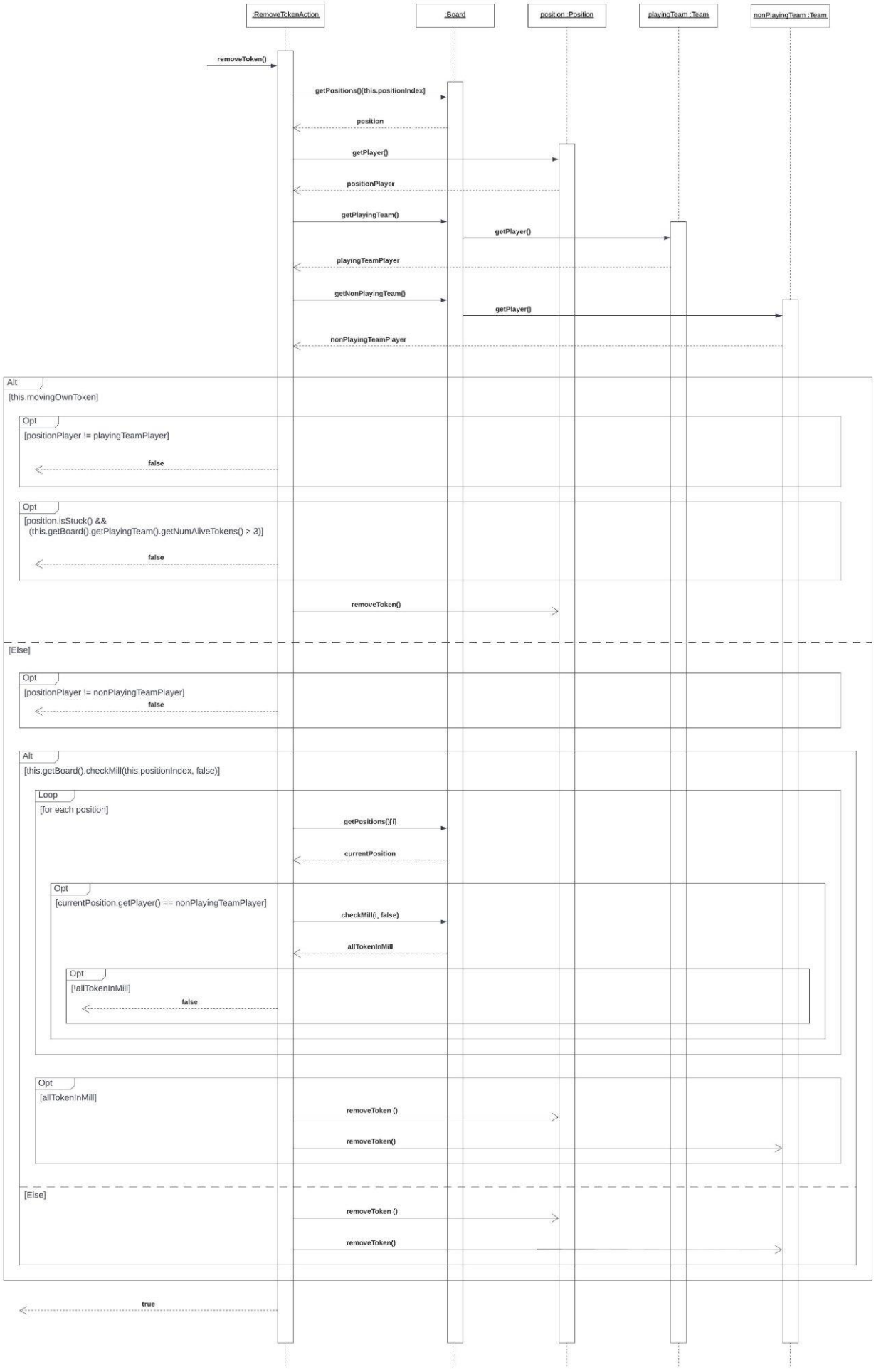
Bootstrap



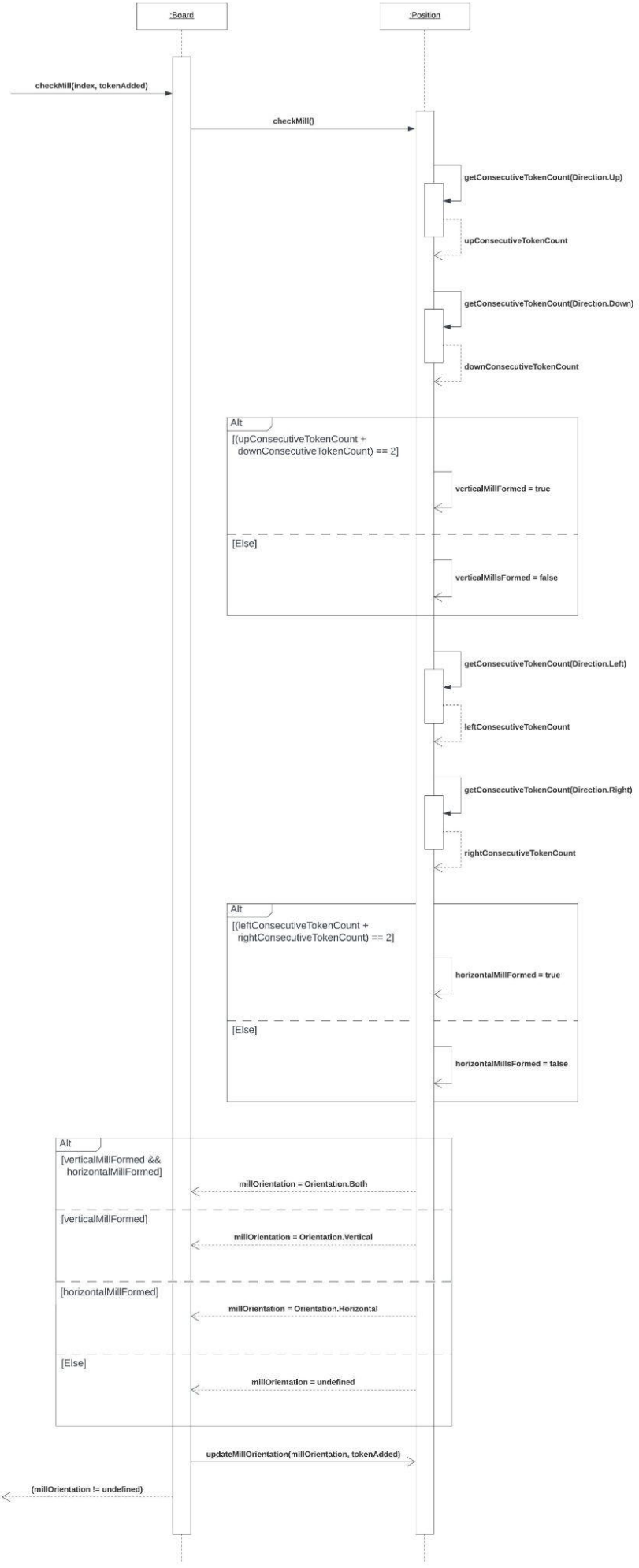
Placing Tokens



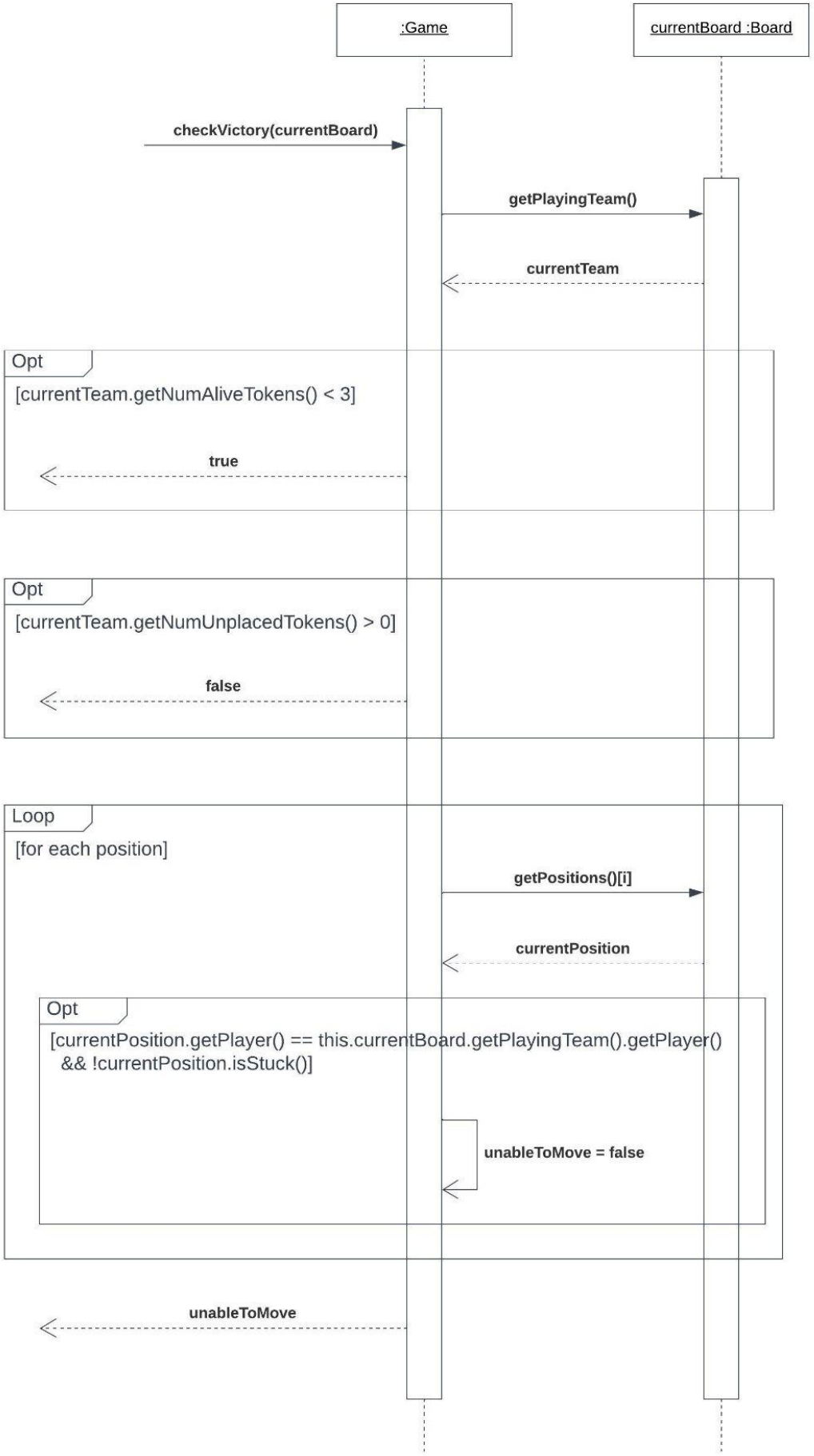
Removing Tokens



Checking Mills



Checking Victory



Architecture

Please note that the architecture below is exactly the same as that of Sprint 2 and has not been revised, as the architecture submitted for Sprint 2 received *Excellent* and no feedback.