FIT3077: Software Engineering: Architecture & Design

Sprint Three Deliverables

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

Table of Contents

Reference Images

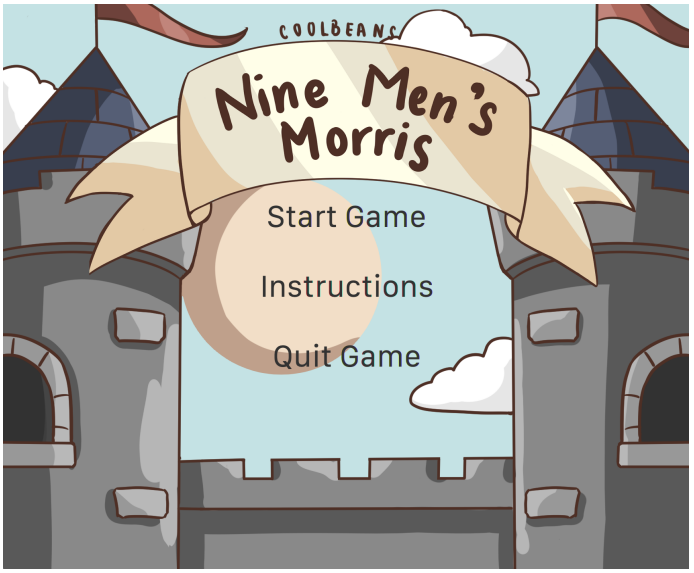


Figure 1: Our team's finalised game title screen design.

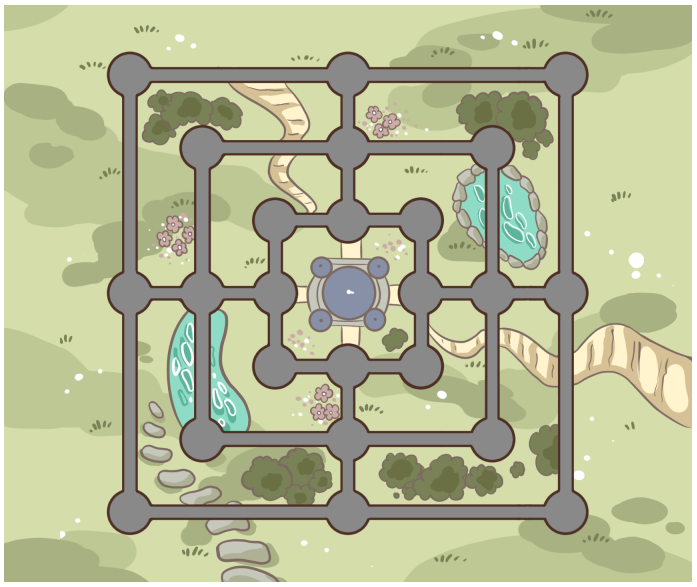


Figure 2: Our team's finalised game board design.

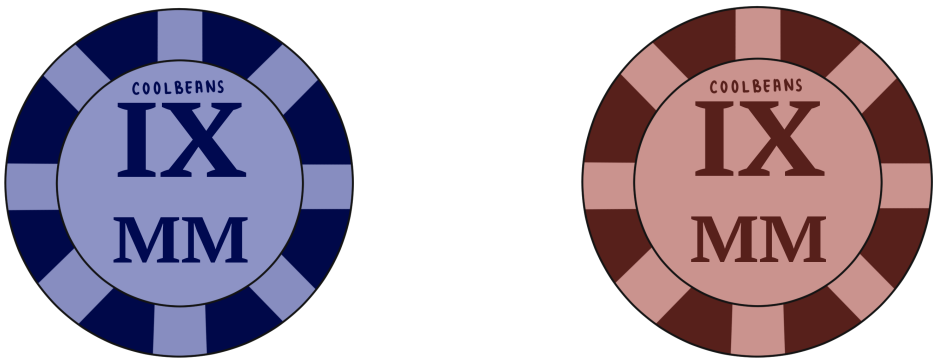


Figure 3: Our team's finalised game token design.

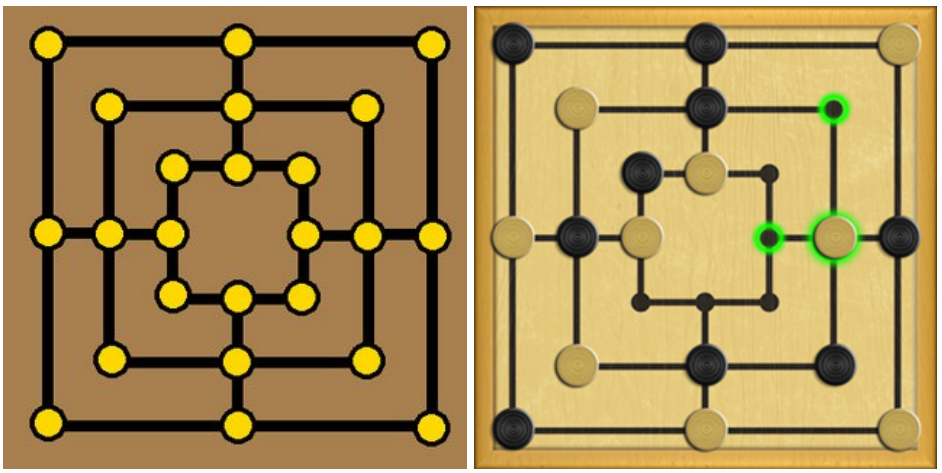


Figure 4: *Typical Nine Men's Morris game board and token design.*

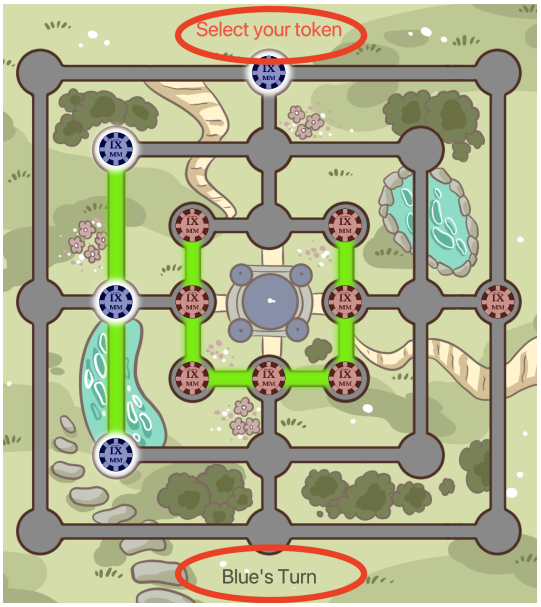


Figure 5: *Image showing feedback on player's move (on top) and turn (at the bottom).*

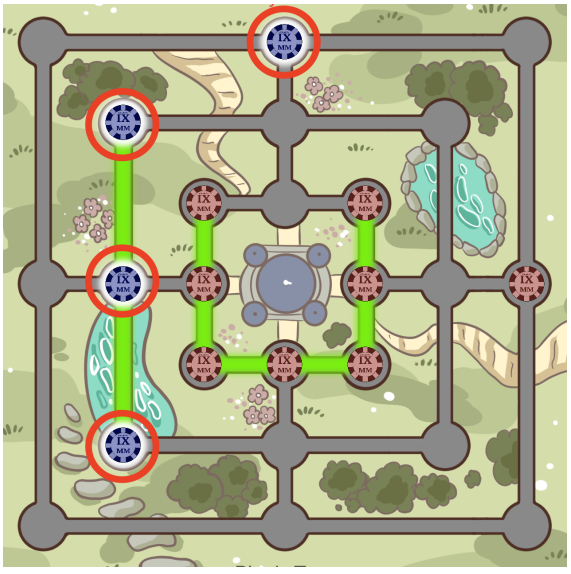


Figure 6: *Image during moving phase. White circles indicate tokens which can be moved.*

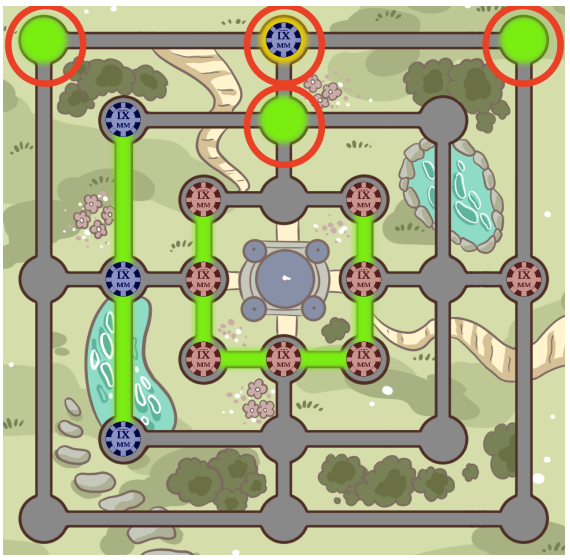


Figure 7: Image during moving phase. Yellow circles indicate the token which has been selected and green circles indicate the destinations the token can be moved to.

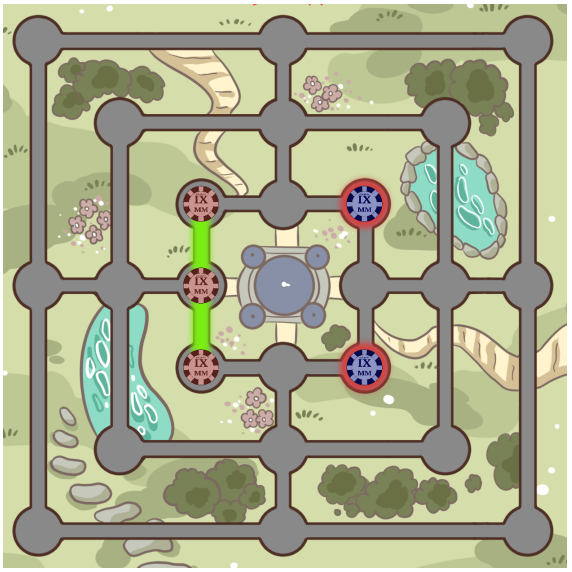


Figure 8: Image after mill is formed. Green lines indicate mills and red circles around tokens show that the token can be removed.

Figure 9: Image showing game results and match counter. Players can navigate back to the title screen menu or play another game.

Figure 10: Image of pause menu. Players can take a breather, restart the game or navigate back to the title screen menu.

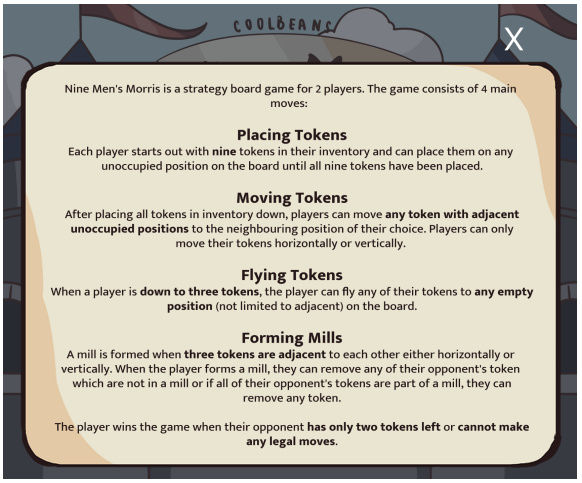
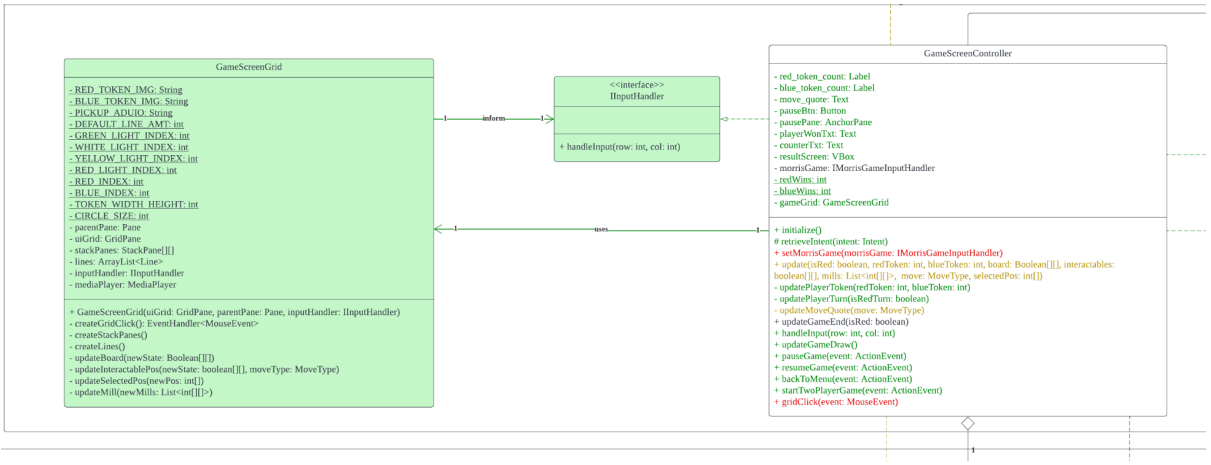


Figure 11: Image of instruction manual which can be accessed through the title screen.

Architecture and Design Rationales

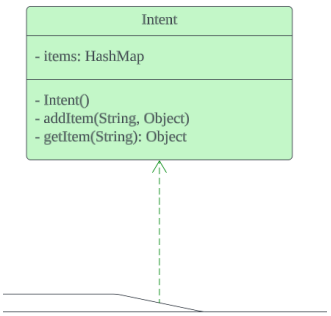
Architecture Revision

GameScreenGrid and IInputHandler



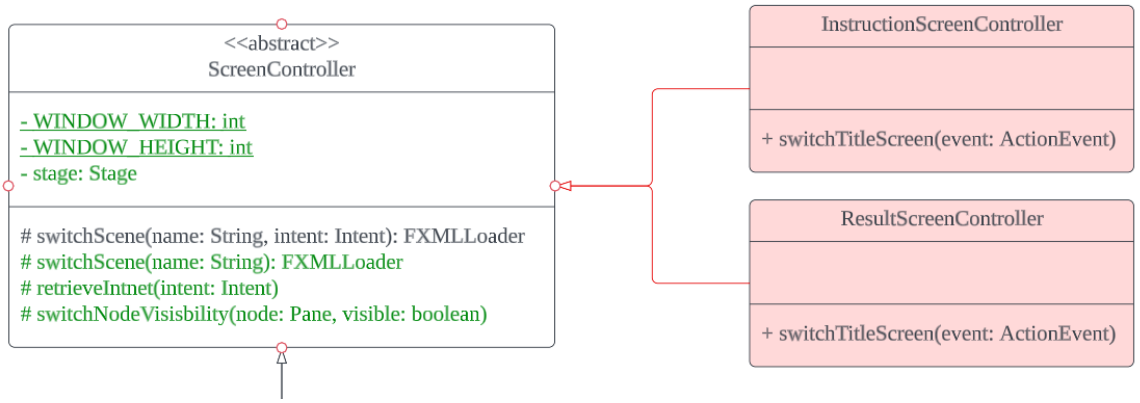
The first architecture revision we made is that we added a new **GameScreenGrid** class which represents the UI grid that the player will interact with in the game and an **IInputHandler** interface which handles the passing the input from the grid UI to the controller. In the original diagram, the GameScreenGrid class and the IInputHandler interface did not exist. This is because the current contents of GameScreenGrid and IInputHandler were all accumulated in the GameScreenController class which we later realised was not ideal. With the addition of GameScreenGrid and IInputHandler, we are now able to separate the game grid logic from the GameScreenController class because the game grid logic can be very complicated and putting them with the controller logic which is also complicated could bloat up the class and decrease its maintainability. By separating the grid logic from the controller, our application abides by the Single Responsibility Principle and ensures that each class is only responsible for their dedicated task.

Intent



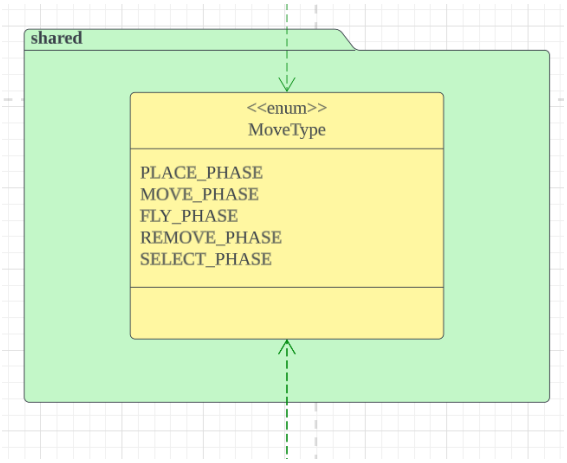
For our next architecture revision, we have also added an **Intent** Class which prepares us for our extension of the game, which is to allow players to play with the AI while reducing the dependency between controller classes. This is because the TitleScreenController needs to let the GameScreenController know that either 2 players are playing against each other or a player is playing with the computer. By having this Intent class, controllers don't need to have a dependency on each other and they get all their information through the Intent class. This follows the open-closed principle as more screens can be easily added and share information with each other without introducing more dependencies.

InstructionScreenController and ResultScreenController



In this sprint, we have also removed the classes **InstructionScreenController** and **ResultScreenController**. This is because our team has found that having separate scenes for these screens is inefficient and decreases our game's performance. Aside from that, our team had also found a better way to show the instructions which is by layering them on top of the title and game screen and toggling its visibility and results since originally, according to the low-fidelity prototype, they were meant to be pop-ups on top of the title screen and game screen respectively.

MoveType



Another revision we made is that we changed the enum **MoveQuote** into enum **MoveType** instead. This is because the original MoveQuote enum that we had previously stored the string constants of the quotes and after more thought on our implementation, our team decided that we needed a more general way to link up the underlying game logic with the display logic which can be done with this enum type. We also decided that the placement of these string quotes needs to be changed if we were to use this enum to link both logics up as having them inside the enum would mesh together both logics again and go against Separation of Concerns. Therefore, we relocated the string quotes for moves into the GameScreenController and renamed the enum from MoveQuote to MoveType to better describe its use. As an effort to organise our files, we had also added a shared folder to indicate files that had a link to both the game logic and display logic and placed this MoveType inside of it.

Addition and Change of Attributes and Methods in MorrisBoard

As with all coding implementation, even though we might be able to plan out our architecture to ensure extensibility, it is quite impossible for many developers to visualise every single possible attribute and method that might be needed for the application to be complete until we actually dip our hands into the water. Hence, in many classes, we would have added the required **attributes** and **methods** along the way as we were coding. Ultimately, this is not too much of an issue as long as the changes we have made are documented properly.

Diagrams

The UML and sequence diagrams that we have created are as follows in the following seven pages. Do take note that due to their size and the limitation of the document size being A4, some things might appear unreadable or arrows might be missing when they are actually there. As a back-up (and in fact, our recommendation), do refer to the links below for high-quality images. For documents in the Google Drive, do download them first for a better viewing experience

Master LucidChart Document: (all individual diagrams can be found at their respective tabs located at the bottom of the page)
https://lucid.app/lucidchart/e1087df5-fc36-4c21-b42d-6ea60f5f69c6/edit?view_items=57lZvbQnF~kc&invitationId=inv_0ad25501-ca73-4852-85d8-a29049b01efc

Bootstrap Process:
GoogleDrive Link:
https://drive.google.com/file/d/1yQJmuoi-ohQfsZTX9FKgiAZYqe3LmtRr/view?usp=sharing
https://drive.google.com/file/d/1L3tWpAOrg7bKFb8OdEoTp-B5Lnexmtwc/view?usp=sharing

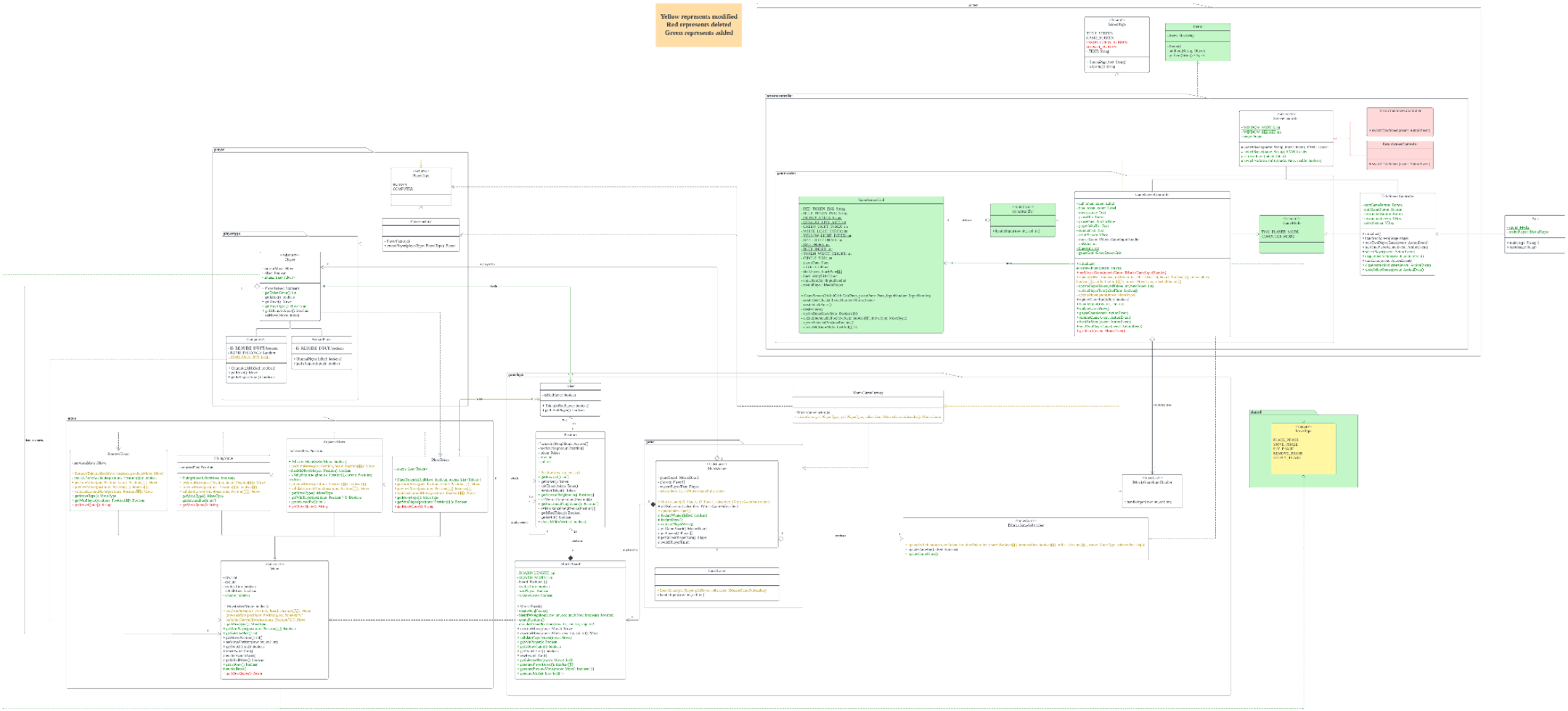
Placing Token:
GoogleDrive Link:
https://drive.google.com/file/d/1CoZ7aeqZ7BNxDzFrWGcUK-IY1sH6FyTC/view?usp=sharing

Moving Token:
GoogleDrive Link:
https://drive.google.com/file/d/1LME7n7j1x_f_bgCmpIkYOspIv3eHkIL9/view?usp=sharing

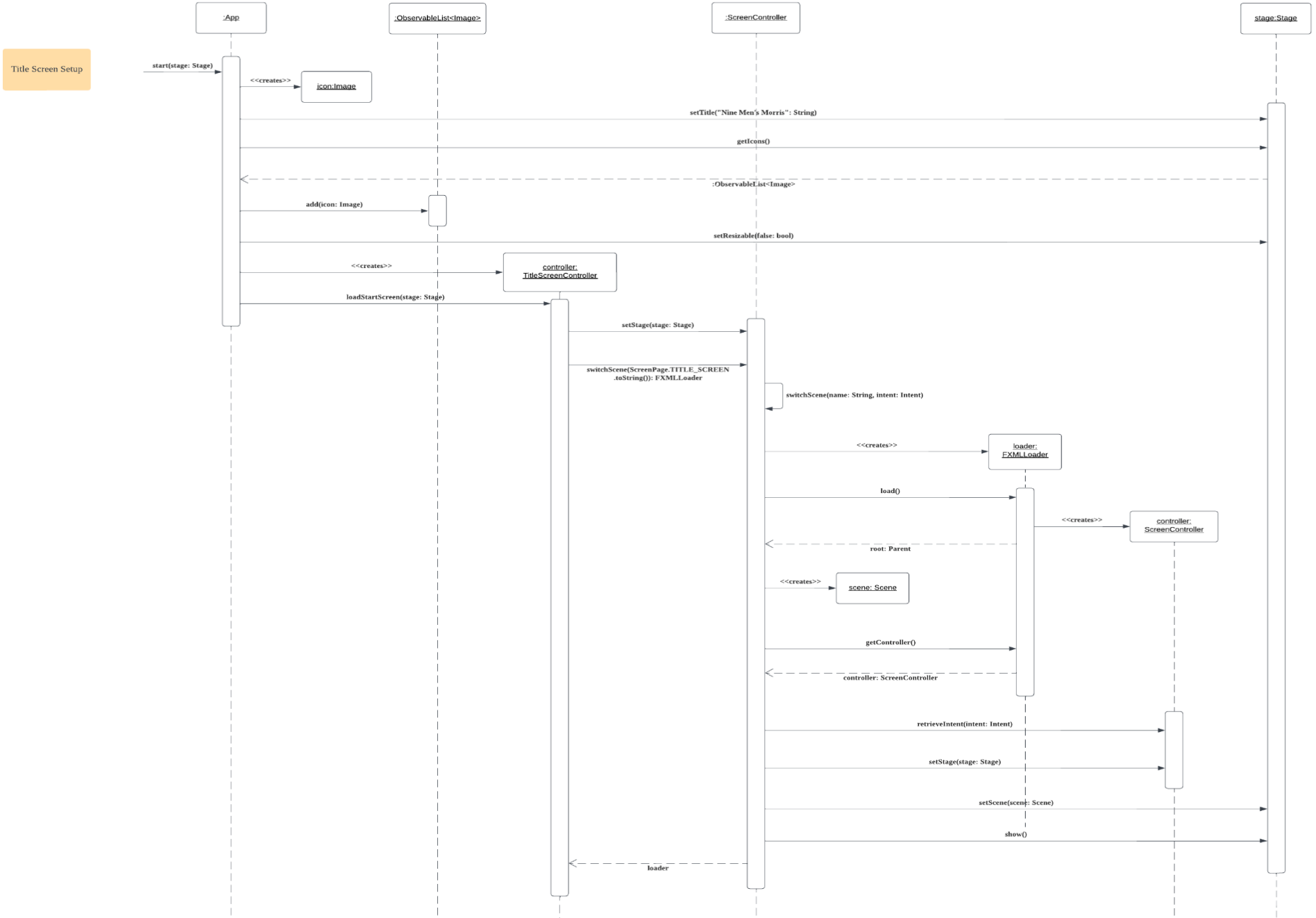
Flying Token:
GoogleDrive Link:
https://drive.google.com/file/d/1-VWnmjYMb_IAG_wk0pvWHIcqrfVFE0ui/view?usp=sharing

Removing Token:
GoogleDrive Link:
https://drive.google.com/file/d/1Eg_IY98dvZKLwW05C-_SQoYJ0oATAHYu/view?usp=sharing

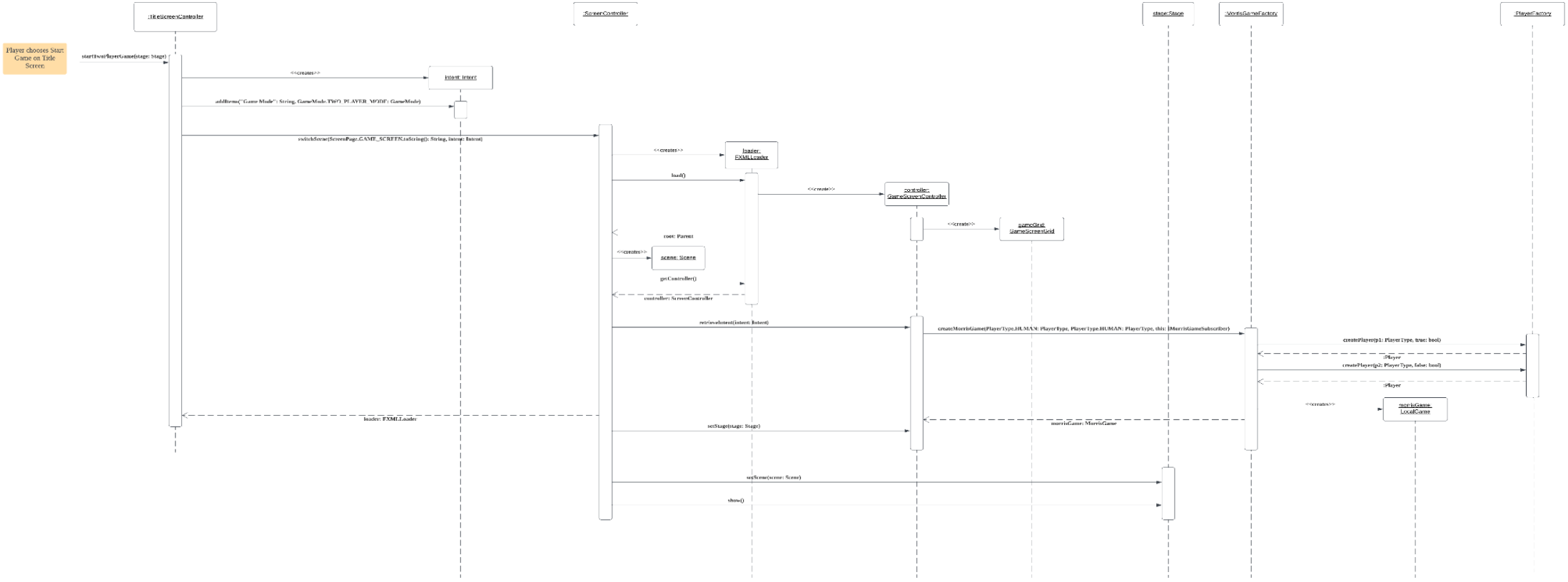
UML Class Diagram



Bootstrap Pt. 1

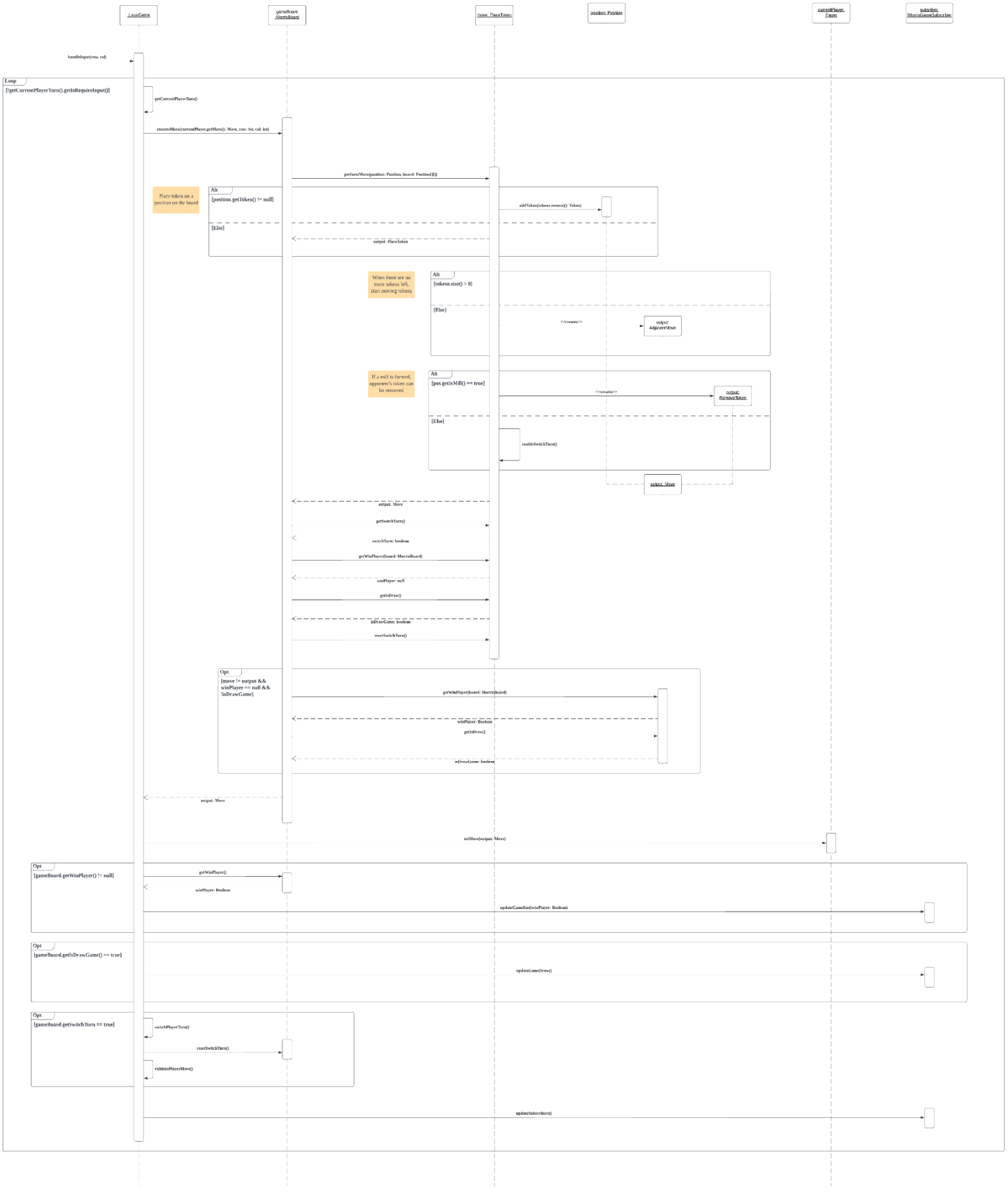


Bootstrap Pt. 2

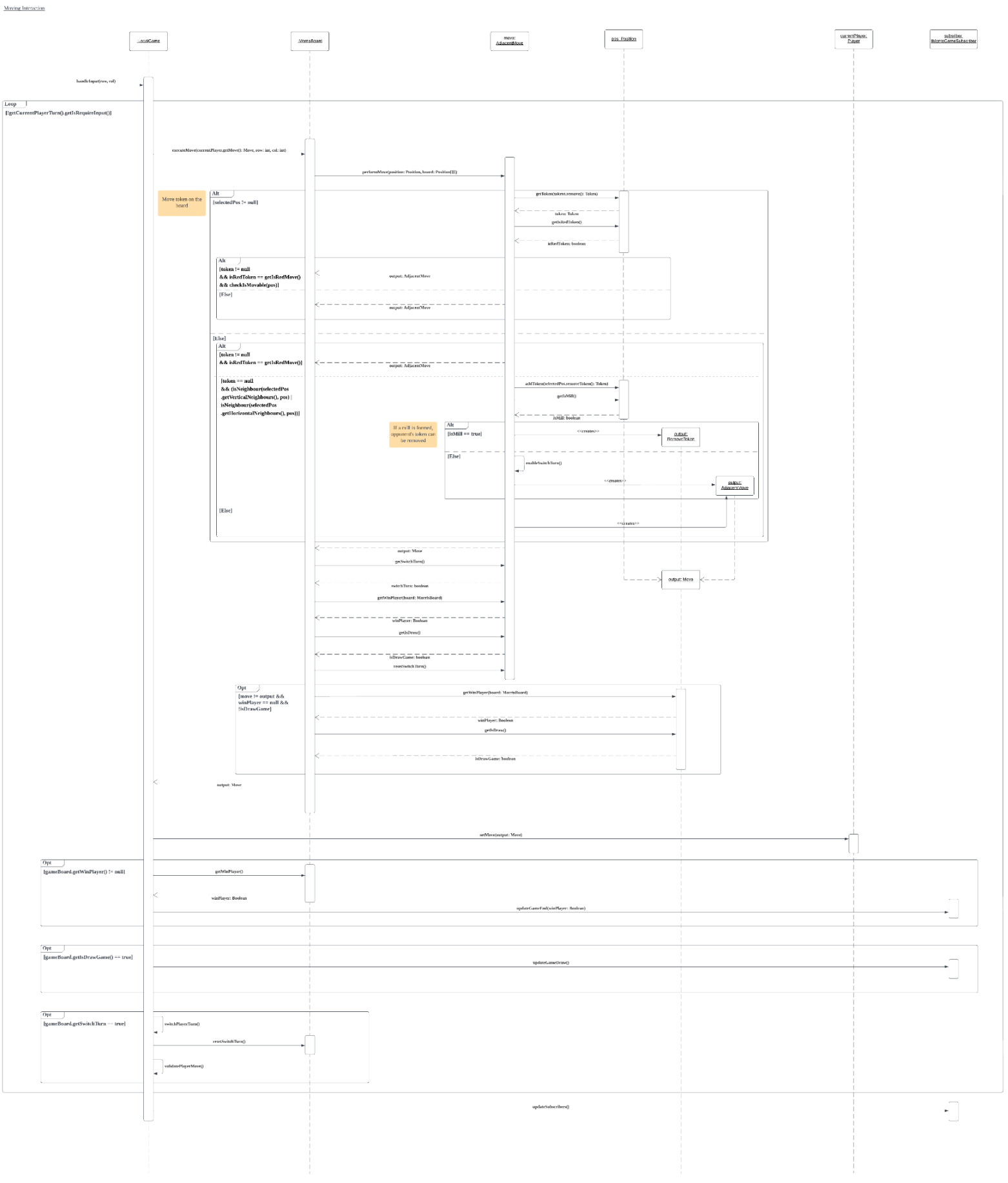


Placing Interaction

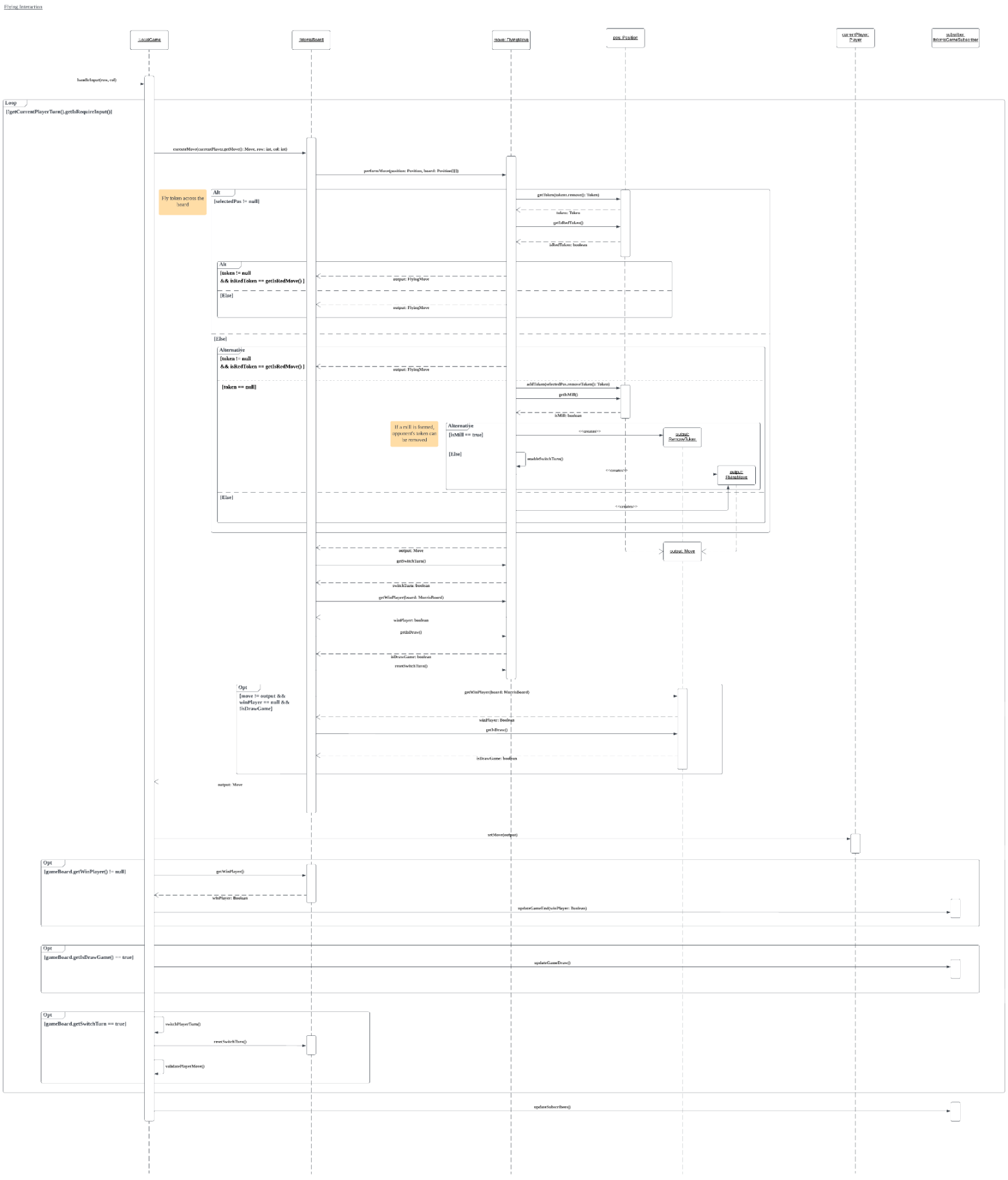
Placing Token



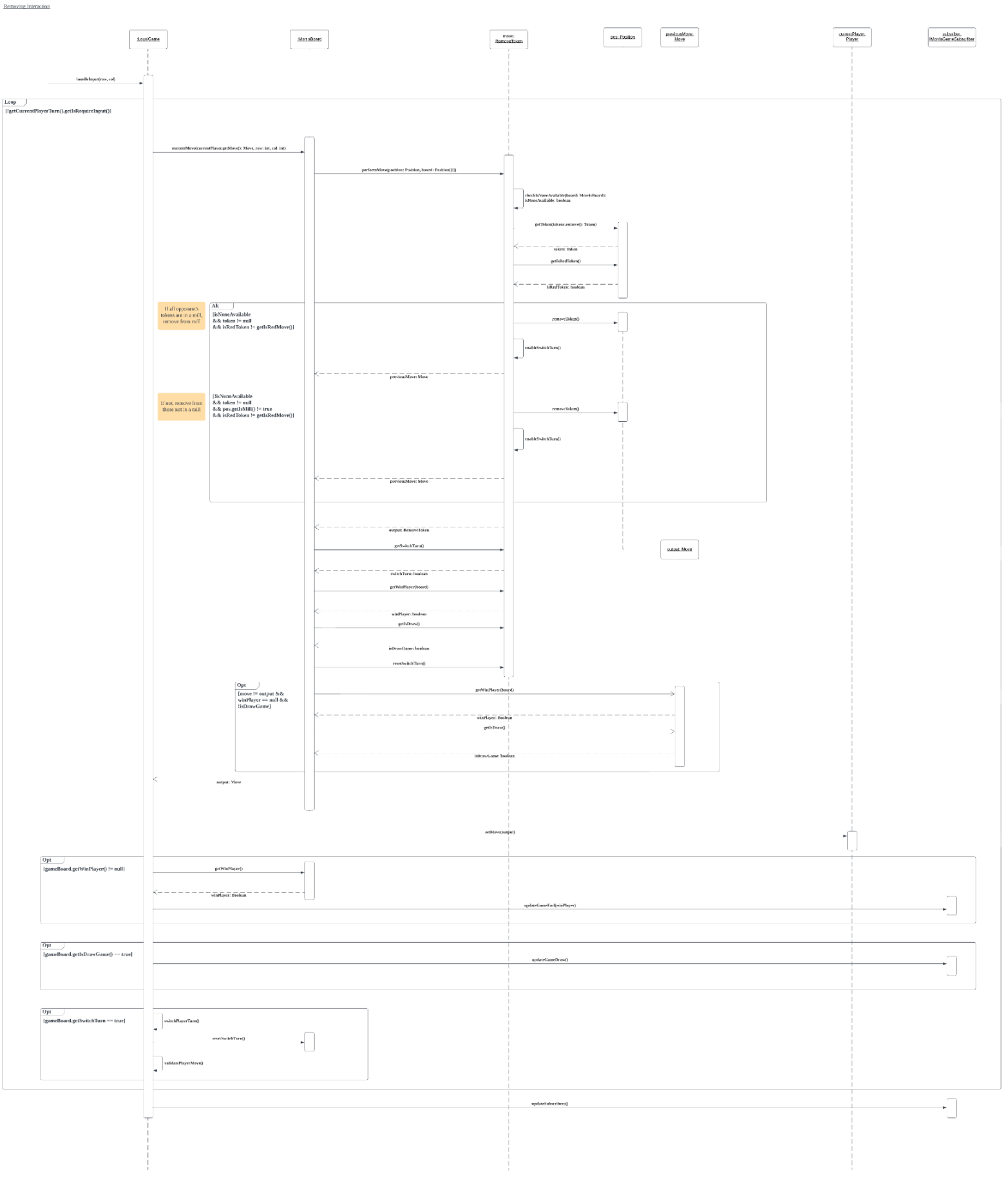
Moving Token



Flying Token



Removing Token



Quality Attributes

Usability

According to both Geeks4Geeks and Nielsen Norman Group, usability can be defined as the extent to how easy it is to learn, operate, and understand the functions of the software. Our software projects this principle throughout the entire game. The main title screen (Figure 1) is reminiscent of those in many basic games including a simple start game button, an instruction button, and a quit game button. Their functionality is self-explanatory and since they are already present in many games, their familiarity is an added advantage in making it easy for our players to operate this screen and promoting usability. Additionally, clicking on the instruction button reveals a manual (Figure 11) which provides guidance to players which would ease the player's thought process when starting the actual game.

The game screen also provides a lot of usability to the user in the sense that in each move, there will be helpful statements that show the type of move that can be made and the current players' turn as well as lights which act as guidance on the moves that the player can make. The lights are also differentiated by colour to show different states in the game. White circular light gives a hint during the moving phase on which tokens can be moved (Figure 6), yellow circular light shows that the token is currently active or selected and green circular light usually provides permission for the player to execute a move (Figure 7). Green linear light shows the position of a mill (Figure 8). This active and responsive feedback which changes very frequently throughout the game reduces the effort that the players would need to remember whose turn it is, know what move can be made and figure out which positions are interactable during the move, improving their gaming experience, ease of use, and allows them to just focus on playing the game instead.

Modifiability

According to Week 6 workshop slides, modifiability is defined as a system that can make reasonable changes at a reasonable cost. While designing our game, we made sure that every component is easily changeable without breaking the game.

There are 3 modifiable components of our game which can be easily changed.

1. The ability to change our local game into an online game with spectators
2. The ability to change all 9-men Morris moves into a different set of rules
3. The ability to change our game screen

For the first modifiable component, our local game can be easily changed into an online game with spectators. Our IMorrisgameSubscriber allows our game to be online such that each online copy of the game is subscribed to our MorrisGame and any changes will be notified to them. This also includes spectators who will be subscribed to our MorrisGame and can see real-time changes in the game.

For the second modifiable component, we can easily change our 9-men Morris moves into a different set of rules. For example, we can change our AdjacentMove class to allow tokens to move 2 spaces instead of 1. This change is only required to be done in the AdjacentMove class and will not require changes for other classes as all validation and operation of the moves are done in the respective Move child classes. Therefore, this also applies to all of the moves in our game which can be easily modified to a different set of rules.

For the last modifiable component of our game, we can easily change our game screen into a different background. We can allow the players to have a settings option to change to the game screen to a background they prefer. Then, we can pass the file name which is a string to the game screen to change the background appropriately. This will not affect the grid lines or token placement as they are all set without the help of the background. Thus, it allows changes in the background of the game screen to be made easily.

Performance

According to the workshop slides in Week 6, performance is defined as the speed the system responds to the user input. With a high performance for the game, the user will not feel that the game is slow and can enjoy the game with more pleasure and satisfaction.

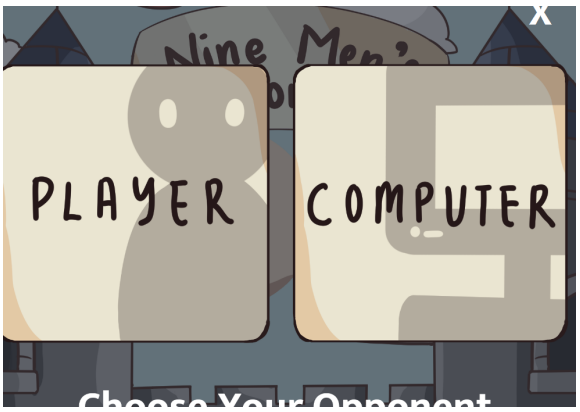
According to Nielsen Norman's group, between 0.1 seconds and 1.0 seconds would let the user feel that the game is reacting instantaneously. To conform to this performance, we would design the game such that each user input would operate in between that time duration.

The timer we decided to design looks like this, where I can call the Timer class and let it start or stop the timer whenever necessary.

```
package ninemanmorris;

public class Timer {

  private double startTime;
  private double endTime;
  private double duration;
  private static Timer instance;

  Timer() {
    startTime = 0;
    endTime = 0;
    duration = 0;
  }

  public static Timer getInstance() {
    if (instance == null) {
      instance = new Timer();
    }
    return instance;
  }

  public void start() {
    startTime = System.nanoTime();
  }

  public void stop() {
    endTime = System.nanoTime();
    duration = (endTime - startTime) / 1000000000; // converting to seconds
    System.out.println(duration);

  }

}
```

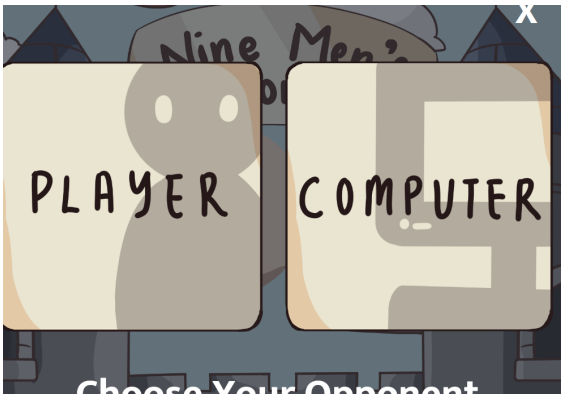
There are 8 situations in which we decided to test the performance of our game as these situations are usually executed by users many times.

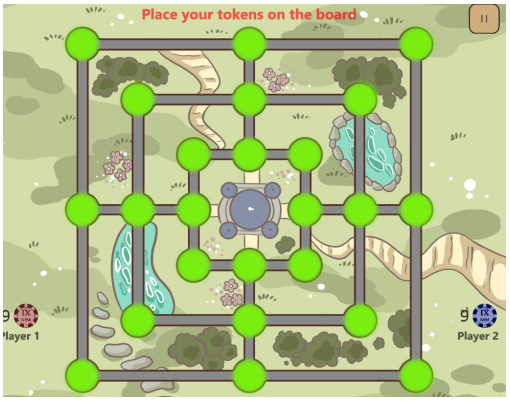
1. Clicking the Start Game button and reaching the Select Player screen



After performing 10 attempts, this operation took an average of 0.0001364 seconds. This is less than 1.0 seconds so it conforms to performance.

2. Clicking the Player button and reaching the Morris board screen



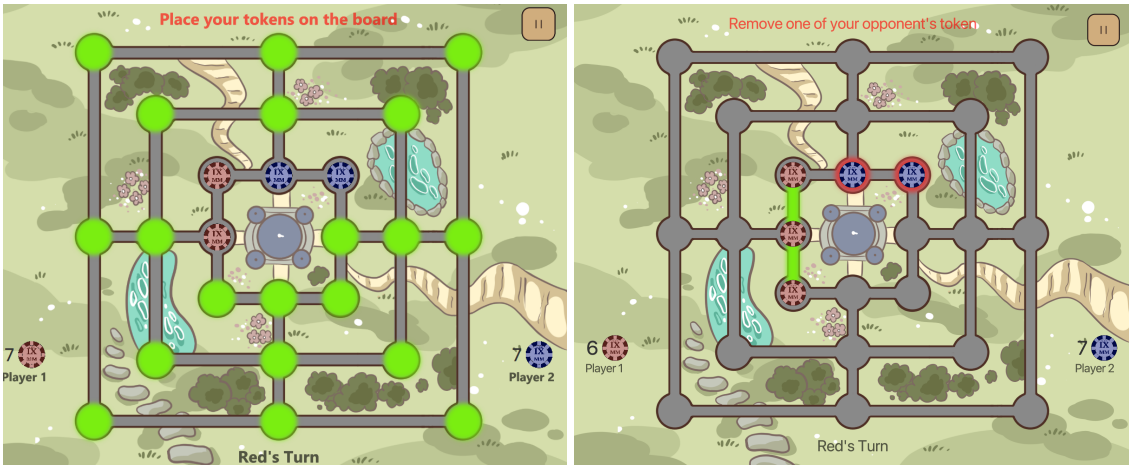


After performing 10 attempts, this operation took an average of 0.3631478 seconds. This is less than 1.0 seconds so it conforms to performance.

3. Placing a token

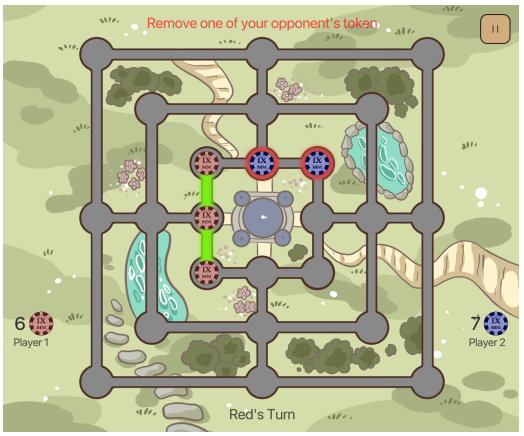
After performing 10 attempts, this operation took an average of 0.0008216 seconds. This is less than 1.0 seconds so it conforms to performance.

4. Detection of new mills



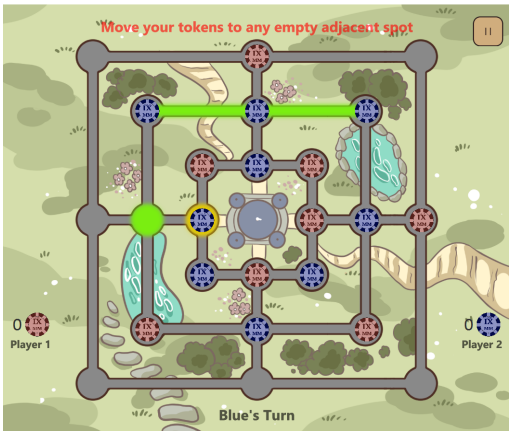
After averaging the 10 attempts, this operation took an average of 0.0000557 seconds. This is less than 1.0 seconds so it conforms to performance.

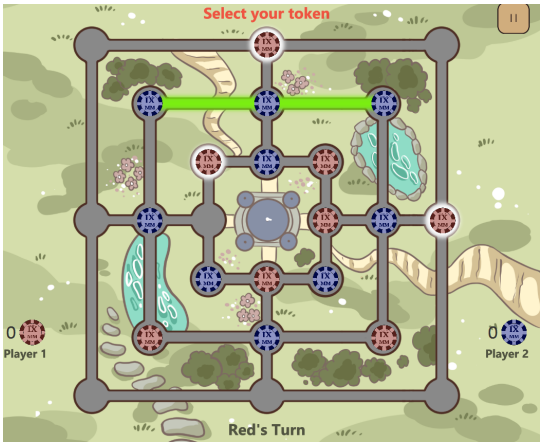
5. Removing a token



After averaging the 10 attempts, this operation took an average of 0.0001871 seconds. This is less than 1.0 seconds so it conforms to performance.

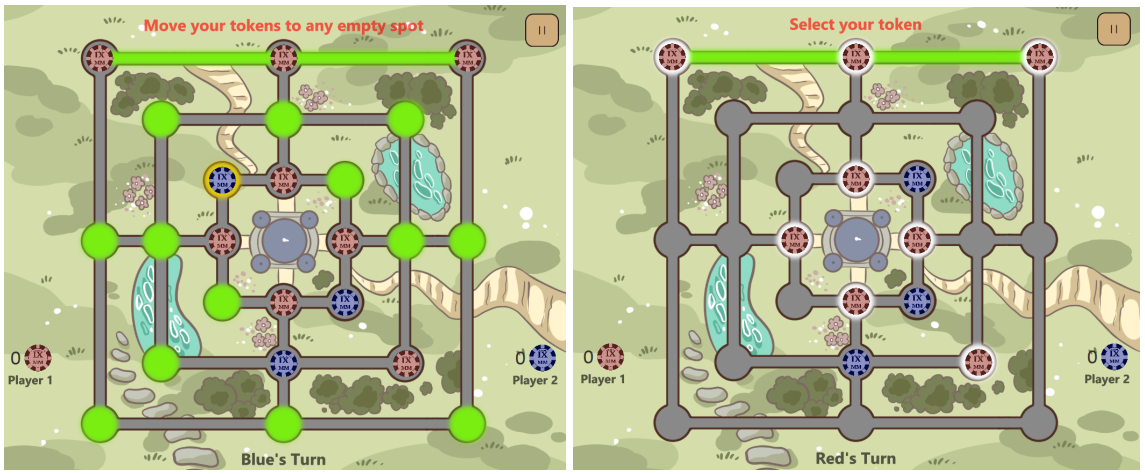
6. Moving a token





After averaging the 10 attempts, this operation took an average of 0.0001399 seconds. This is less than 1.0 seconds so it conforms to performance.

7. Flying token



After averaging the 10 attempts, this operation took an average of 0.0001058 seconds. This is less than 1.0 seconds so it conforms to performance.

8. Detection of end game (caused when a player has lesser than 3 tokens)

After averaging the 10 attempts, this operation took an average of 0.0003616 seconds. This is less than 1.0 seconds so it conforms to performance.

After conducting these test performances, all situations have a time duration lesser than 1 second and the user will perceive these actions as instantaneous. Thus, our application conforms to performance.

Human Values

Respect for Tradition

Before our team has even come up with the low-fidelity prototype for our game, we have conducted some research into the history of the game itself. After doing research, our team discovered that the game itself is very old and dates back to the Roman Empire. The game was popular among Roman soldiers who likely introduced this game to central Europe and England when spreading the empire where it was very popular during medieval times. The game was played by a large variety of audiences, including tavern-goers and children. With all of this in mind, we came up with the design for our prototype:

1. The design of the title screen (Figure 1) and the board (Figure 2) resembles a castle to pay tribute to the Roman soldiers who popularised the game while expanding their empire. This together with the background music caters and gives a medieval feel to the game.

2. The token design (Figure 3) resembles that of a poker chip which would appeal to tavern-goers as poker is a popular game to be played while enjoying the night off with buddies in the tavern over some booze.

3. The overall user interface design particularly the colour choices were picked to provide a friendly feel to the game. The colour palette chosen is mostly bright, pastel and desaturated colours while still including different hues which would be appealing to both children and adult players alike.

We believe that the prior research we have conducted together with the incorporations of the research into our design shows that our team has definitely practised respect for tradition in our software design.

Creativity

At first glance, it is already obvious that our team's board and token design deviates largely from what is usually used as a Nine Men's Morris board while still being familiar enough to both experienced players of the game by retaining the overall shape of the original board game.

Normally, a Nine Men's Morris board would only consist of solid lines and circles and tokens would only be mono-coloured (Figure 4). However, as people who enjoy playing games in their past time as stated in our team member details in our first sprint, our team appreciates and prioritises game aesthetics and has taken the liberty to revamp the design of both the game board and the token with custom-designed graphics so that we can elevate the normal Nine Men's Morris game into something wonderfully different that can provide a more enjoyable game experience for our audiences. This shows that our team has incorporated creativity into our software design.

Pleasure

As mentioned in our previous section, we are game enthusiasts who wish to bring enjoyment and pleasure to our game's players because the more pleasure you have while playing a game, the more likely you are to come back to it. For a pleasurable game experience, some crucial factors to consider are that the game must be simple enough to figure out without causing frustration, able to engage the player in gameplay, considers the player's feelings and allows room for error because different players are bound to have different mindsets in approaching the game. To ensure this goal is achieved, our game includes:

1. Immediate feedback on the player's turn and moves (Figure 5) which can be made during the game itself which serve to inform the user on the current happenings of the game so players are not forced to actively remember.

2. Helpful game lighting (Figure 6, Figure 7, Figure 8) which minimises the cognitive load of the player and allows the player to visualise and plan out their strategy more effectively before making a move on their opponent.

3. Excitement in declaring the winner of the game through exclamation and keeping track of the matches won between players (Figure 9). This gives the winning player a sense that their achievement is recognised while motivating both players to be competitive and continue playing with the addition of the match counter.

4. Pause menu (Figure 10) for the player to restart the game or navigate to the title screen. Adding this shows that our team cares and considers the player's feelings and usability by due to the error handling obvious in the restart button so players can easily restart another game from their current one if a blunder or mistake has been made and the ability to return to the title screen to view the instructions of the game is also a helpful addition.

5. Background music which keeps the player engaged in the game and in some situations might intensify the atmosphere of the game when the crescendo of the tune kicks in.

References

City of Pickering. (n.d.). Nine Men's Morris. *Nine Men's Morris*. https://www.pickering.ca/en/discovering/resources/NineMensMorrisRules.pdf

Nine Men Morris Game. (2021, March 21). The Hyland House Museum. https://hylandhouse.wordpress.com/at-home-activities/nine-men-morris-game/

GeeksforGeeks. (2022b). McCall's Quality Model. *GeeksforGeeks*. https://www.geeksforgeeks.org/mccalls-quality-model/

Usability 101: Introduction to Usability. (n.d.). Nielsen Norman Group. https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Response Time Limits: Article by Jakob Nielsen. (n.d.). Nielsen Norman Group. https://www.nngroup.com/articles/response-times-3-important-limits/

Bell, R. C., B., M., & C. S., F. R. (1969). *Board and Table Games from many Civilizations*. Oxford University Press. https://ia601405.us.archive.org/8/items/B-001-002-771/B-001-002-771.pdf

Image Source: https://www.google.com/search?q=nine+mens+morris&source=lnms&tbm=isch&sa=X&ved=2ahUKEwi1urTG8_n-AhU77TgGHeYdBaUQ_AUoAXoECAIQAw&biw=1309&bih=639&dpr=2.2#imgrc=o10UKr_4sNW5WM

Image Source: https://www.google.com/search?q=nine+mens+morris&source=lnms&tbm=isch&sa=X&ved=2ahUKEwi1urTG8_n-AhU77TgGHeYdBaUQ_AUoAXoECAIQAw&biw=1309&bih=639&dpr=2.2#imgrc=UHCZvUP8_zd0BM