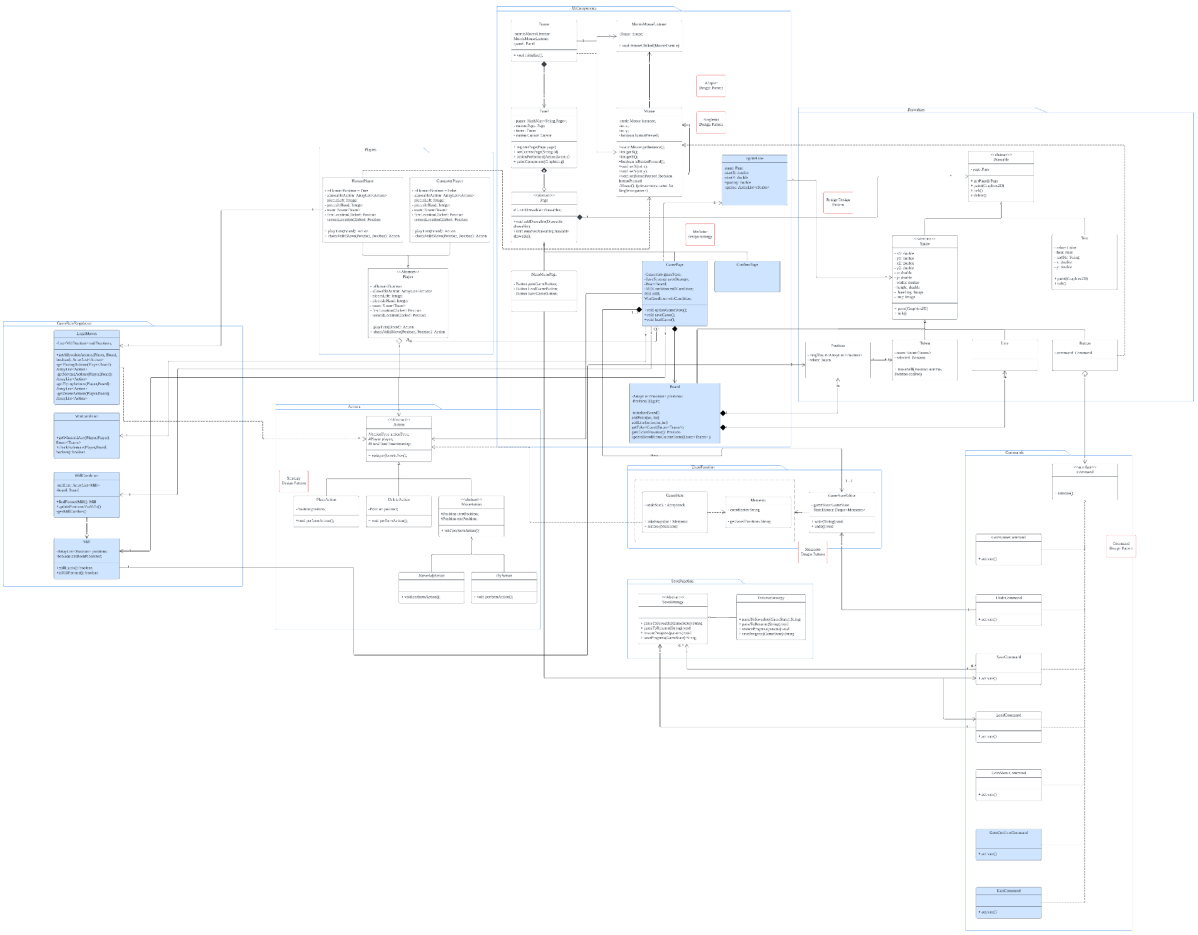
FIT 3077 Assignment (Sprint 2): 9 Men's Morris Game Application

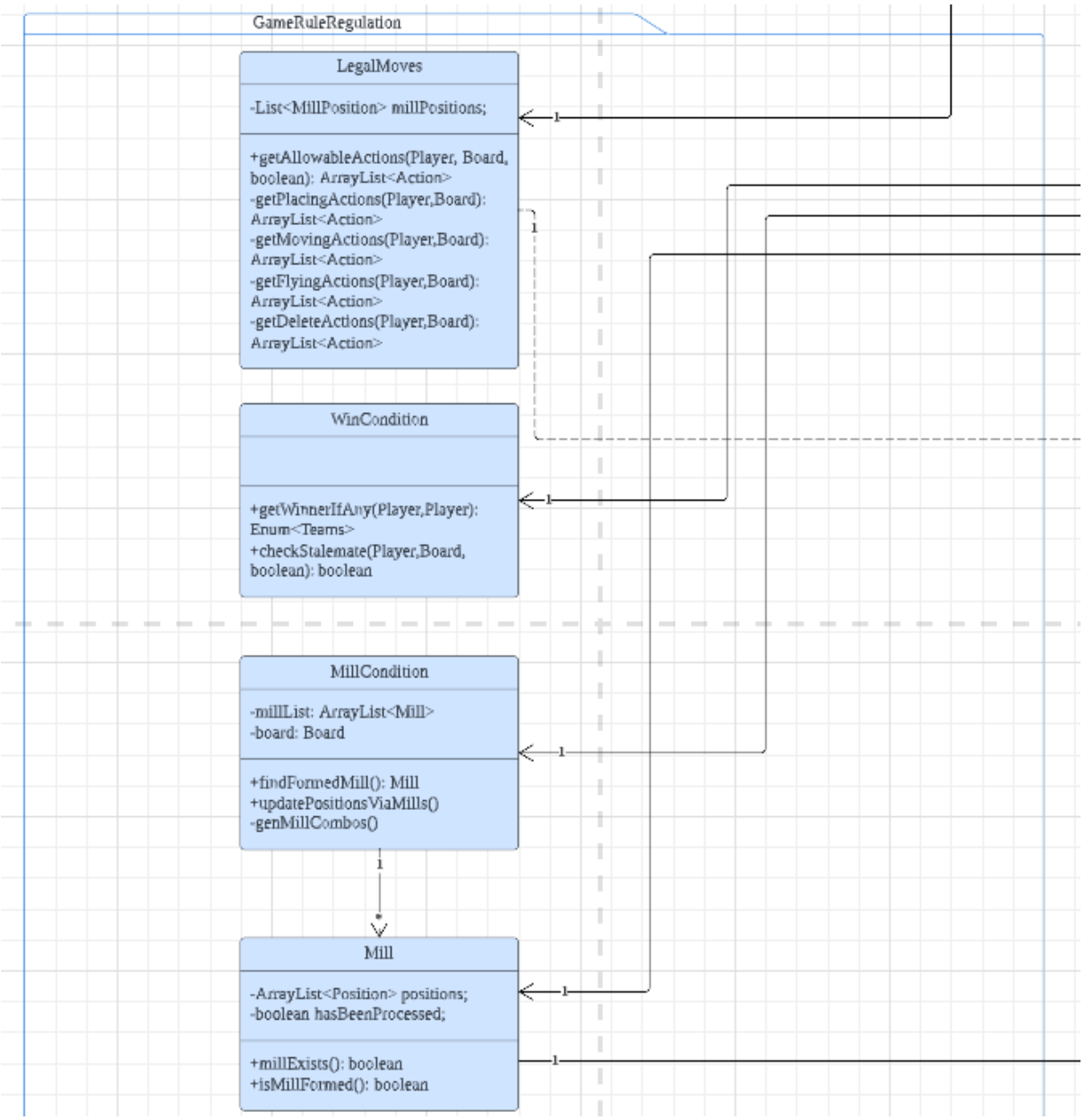
Presented by:

Ong Chien Ming - 31861318
Syed Zubin Hafiz - 3227671
Garret Yong Shern Min - 31862616
Mario Susanto - 31103146



Changes to Class Diagrams

Game Rule Regulations

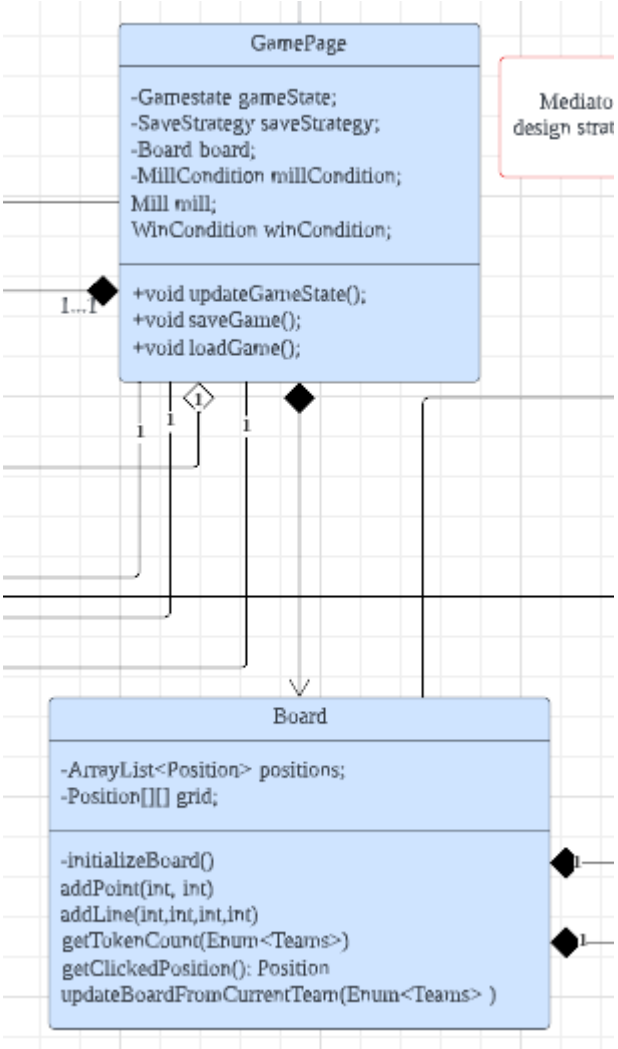


The classes under the GameRuleRegulations package have been modified extensively, including the removal of the GameRule class and the addition of the Mill class. It was necessary to remove the GameRule class as we realized that the GameRule's subclasses have nothing in common with each other, and that there are no benefits to polymorphism under the circumstances in which the WinCondition, MillCondition, and LegalMoves classes can be used within. As such, we decided to turn them into independent classes.

We also decided to add the Mill class, as we realized that there was information and functionality that are relevant to a single mill specifically, such as storing the positions

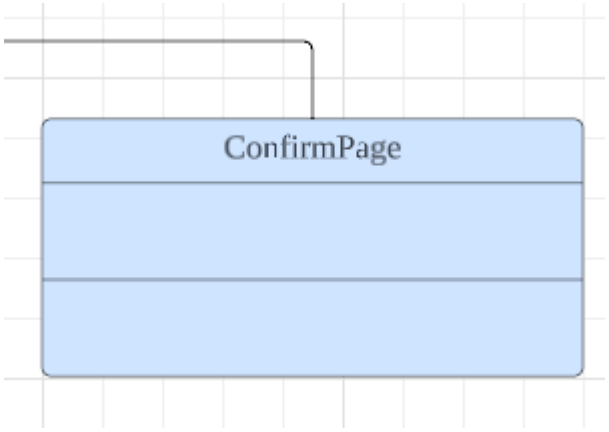
associated with a mill. As such, we needed to add a class to handle this, along with an association from the Mill class to the Position class.

Game Page Board



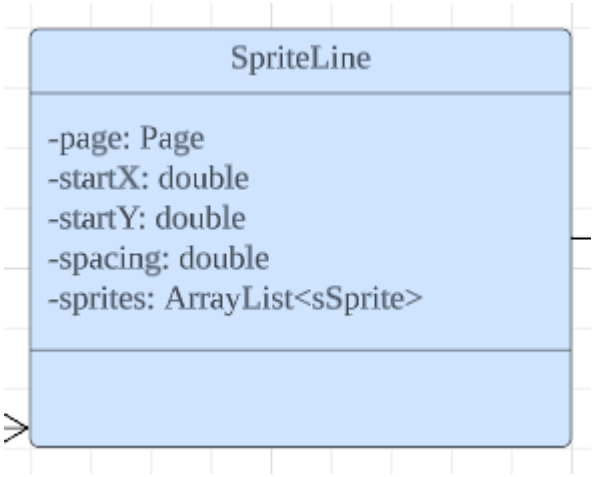
Both the GamePage and Board classes were updated to account for functionality that was not initially considered, including a getClickedPosition() and a getTokenCount() method in the Board class, along with multiple additional attributes in GamePage. These changes were necessary as we had a better understanding on how to better implement the overall game, leading to changes that had to be made to these classes, including exposing more information both internally and externally.

Confirm Page



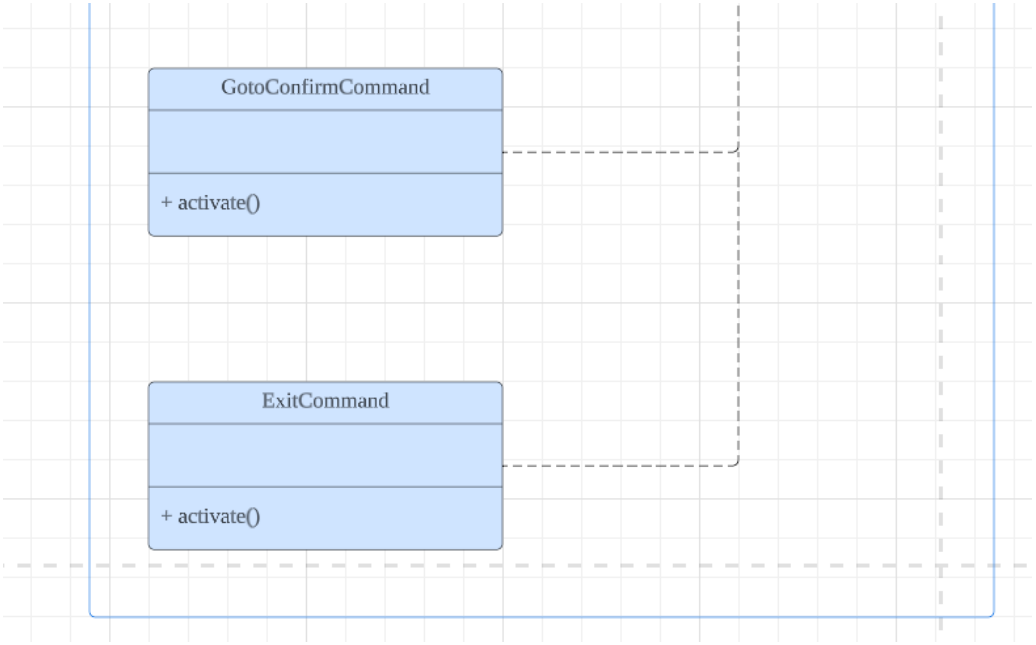
In order to improve the usability of our game by avoiding the possibility of someone accidentally exiting to the main menu, we decided to add a page to confirm whether or not the player intended to quit the game. This therefore prevents frustration from a user losing their game progress.

Sprite Line



Whilst modifying the application's UI to better correspond with the low-fi prototype, we realized that we needed some method for representing the number of tokens each player has not placed yet graphically rather than via a simple text. While we could have had a method in GamePage to place these tokens, we decided to instead have a separate SpriteLine class to handle this behaviour. This ensures a separation of concerns, improving maintainability, while also reducing the likelihood of GamePage turning into a god class.

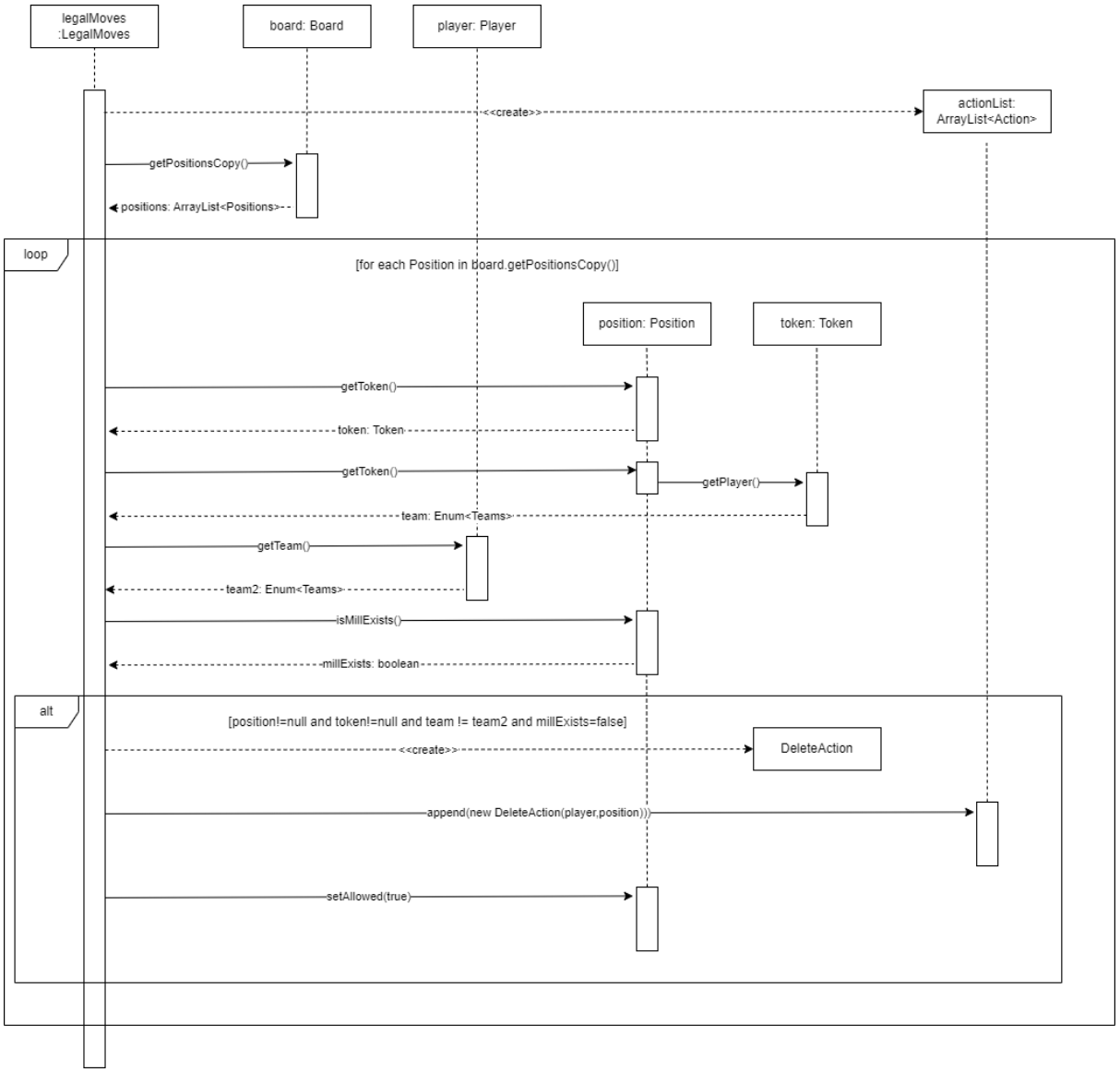
Exit Command and Goto Confirm Command



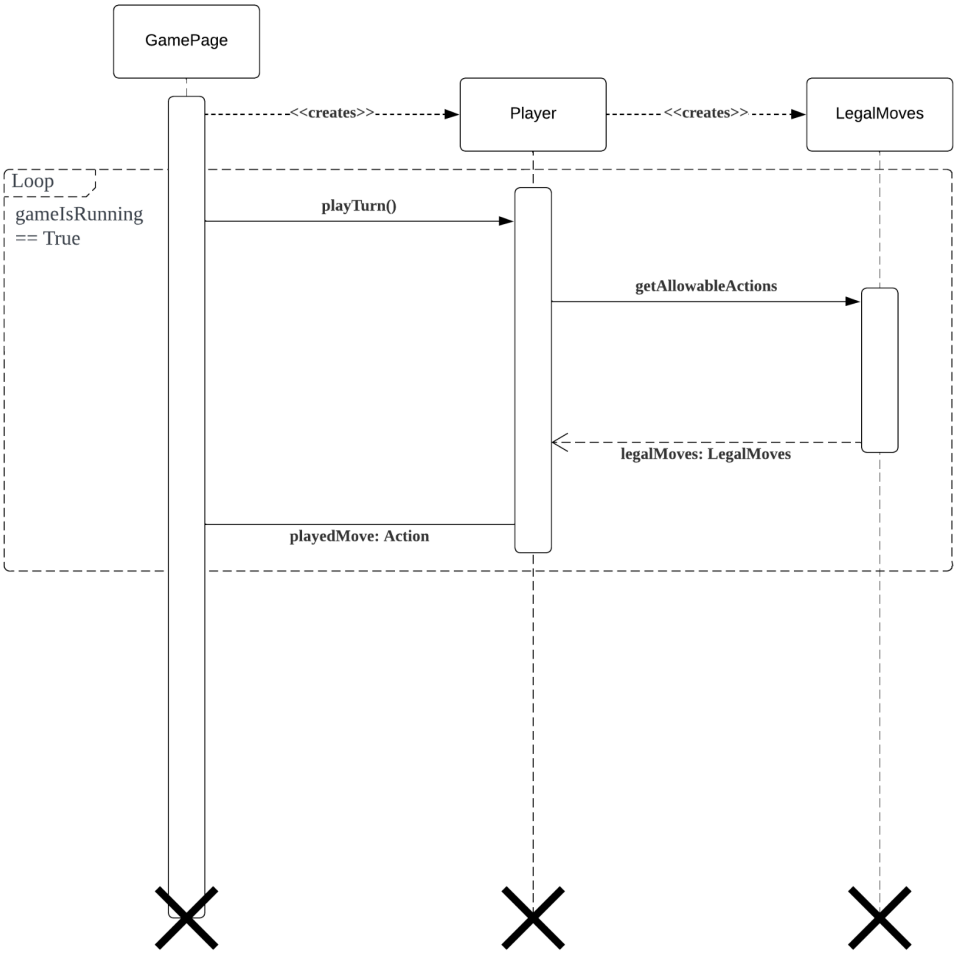
While developing the game, we decided that we needed additional commands for buttons for some minor functionalities that were not initially considered. The GotoConfirmCommand class is needed in order to be able to add a button on the game page that opens a prompt for the user to either continue playing or exit to the main menu. Meanwhile, the ExitCommand command closes the window of the application, and is needed for the exit button on the main menu.

Sequence Diagrams

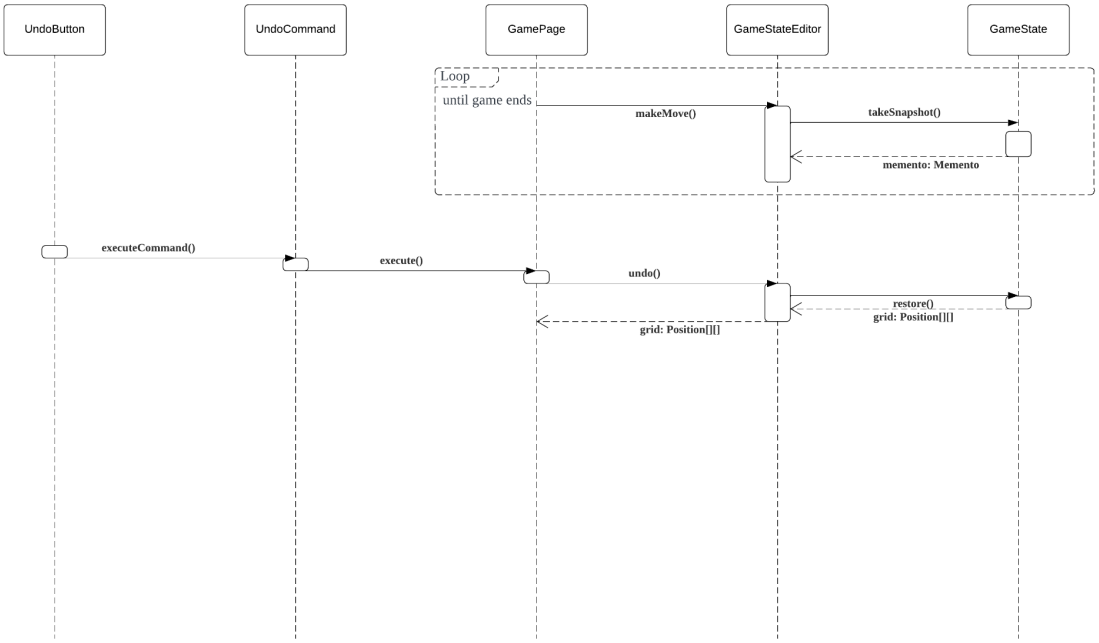
<1> Obtaining a list of actions to delete a token



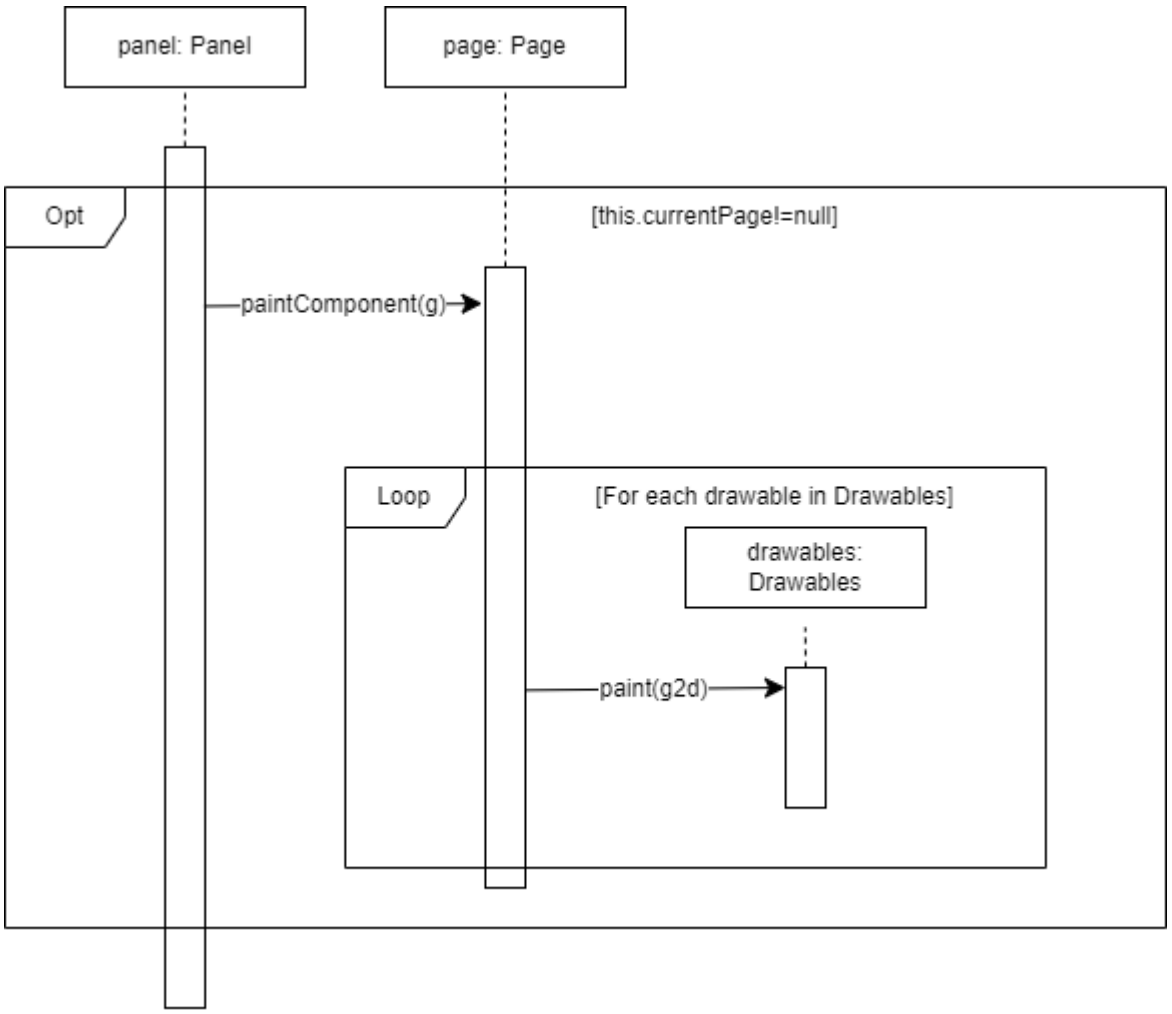
Player making turn



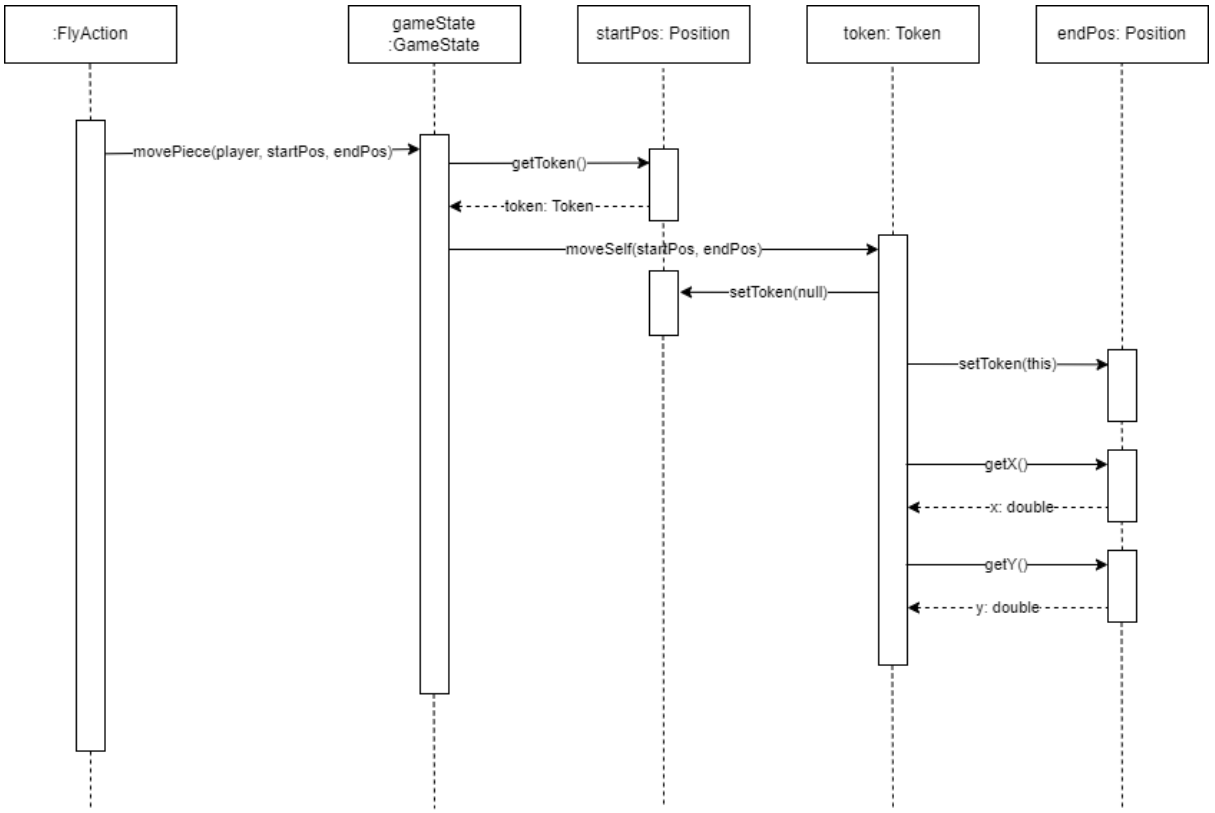
Process of Undoing



<3> Displaying the drawables


panel: Panel
page: Page
Opt
[this.currentPage!=null]
paintComponent(g)
Loop
[For each drawable in Drawables]
drawables:
Drawables
paint(g2d)

<4> Executing the fly action



Non-Functional Requirements

<1> Accessibility:

The game is designed with a focus on accessibility, adhering to established accessibility standards and guidelines. As such, throughout the design process of the game, accessibility was taken into account. One feature that was designed with accessibility in mind was the design of the tokens. This is because we ensured that the tokens were not only coloured differently, with ducks having darker colour tones than geese, but also are shaped differently, thus significantly reducing the likelihood that a person with visual impairments would struggle with distinguishing the tokens. Furthermore, we ensured that background objects and foreground objects had clear contrasts to avoid difficulties in distinguishing them. Finally, when selecting the font that we were to use, we took care to choose a font that was clear and easy to read. By implementing these accessibility measures, our aim is to create an inclusive and enjoyable gaming experience for all users.

<2> Usability

Explain 2-3 quality attributes (as non-functional requirements, e.g. usability, flexibility) that you consider relevant to the 9MM game and have explicitly considered in your design. Why are they relevant and important to your game? Show (provide evidence) how your design manifests these non-functional requirements.

Usability is a common term when it comes to designing an application and plays a significant role in ensuring the application developed receives the right amount of recognition. Usability is an Non-Functional Requirement (NFR) and a term to “measure” if an application is easy to use, easy to understand, easy on the eyes and overall, **usable**.

When it comes to usability, there are no strict rules or guidelines to follow, instead there are design patterns that have been tried and tested to ensure higher levels of usability. For this analysis we are going to be implementing **Don Norman’s 7 Design Principles** & **Ben Schneiderman’s 8 Golden Rules**

These rules cover the following:

1) Visibility
2) Feedback
7) Signifiers
8) Design Dialogues to yield closure

3) Constraints
4) Mapping
5) Consistency
6) Affordance
9) Prevent Erros
10) Permit Easy Reversal of Actions
11) Reduce Short-term memory load

When designing our application we took numerous of these rules into consideration. Due to some of the features that have not been implemented yet, it is not possible to observe some of these rules.

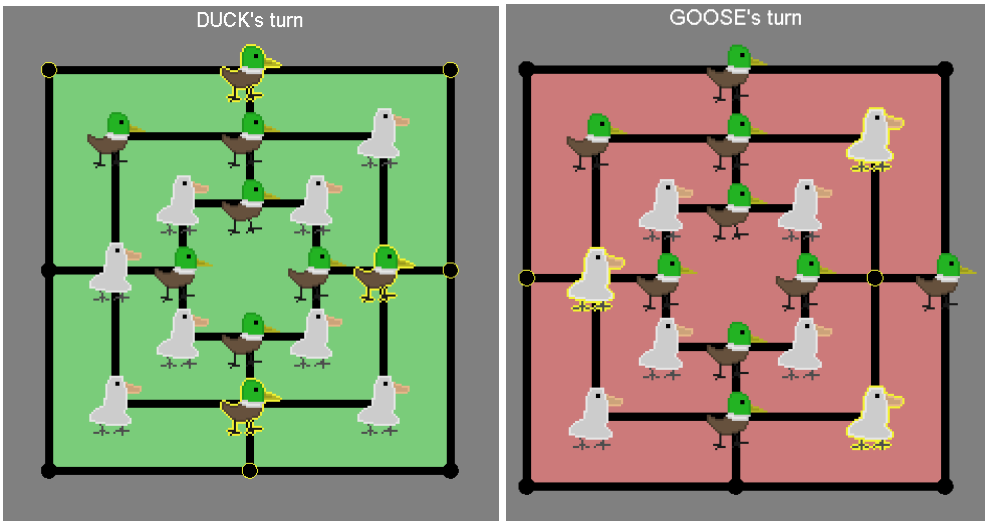
Visibility can be observed in the application where all the options are clearly listed for the users to pick and all the buttons are labelled clearly. Besides that, the elements in our application are packed neatly and the elements are not cluttered.
(insert updated menu screenshot here)

Feedback can also be observed all throughout the application. For instance, when selecting a piece to move, the piece can be visibly greyed out to reassure the player that the piece that they have clicked on has been selected.

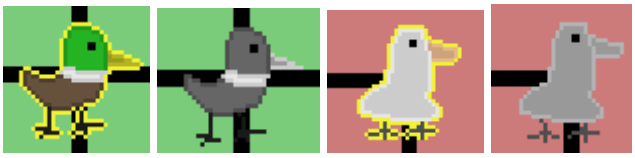
Constraints can be observed while playing the game; we have implemented it so that the players are only allowed to make legal moves. For instance, once you start the game (during placing phase), you are allowed to click on any of the points on the board to place a piece but once you reach the moving phase (no more pieces to place and have to move the pieces) you are only allowed to move the pieces on your team and move them to legal spots on the board.

Mapping cannot be observed in the application as there are no features that explicitly make use of this rule. However, that being said, there are no violations to this rule.

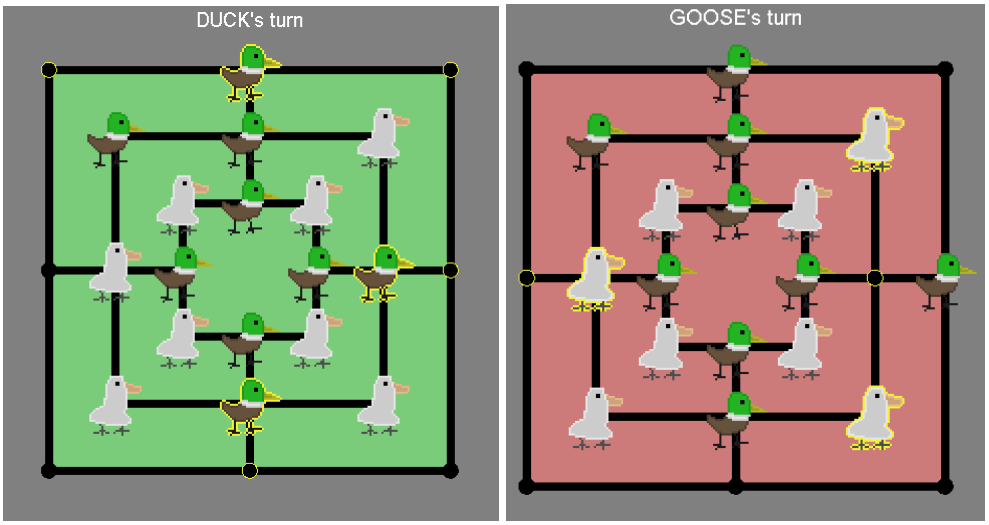
Consistency can be observed within the game. There are many major and minor displays of this rule; for instance, when it is a players turn all valid pieces to be moved are highlighted in yellow for both teams as shown below



To add on to the previous example, when a player selects a valid piece, the piece would turn grey for both teams. Evidence is as shown below:

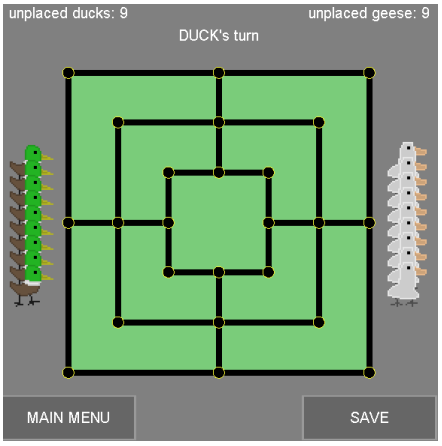


Affordance is another rule that can be observed in our application. We try to ensure that we are able to accommodate to as wide a user base as possible. While there is very little to nothing we can do for the blind community, our application does well to help with users with colour blindness. For instance, the colour on the background of the board is used to denote which team is currently having their turn, the background use different colours (the colours highly contrast with each other) for each team to show this.



However, in case the user has monochromatic vision, there are words in the top-middle of the screen that explicitly say whose turn it is.

Signifiers is very prominent in the application as for every button in the application has names or rather labels that tell the user what each button is supposed to do. This communicates how to use the design to the user(s).

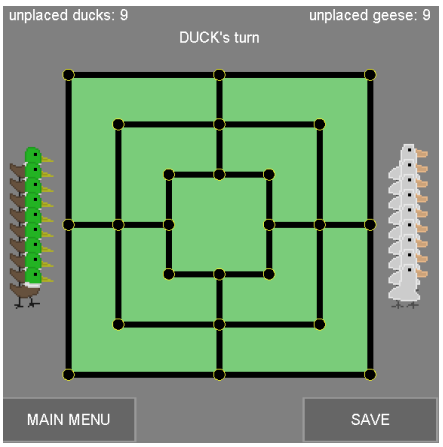


Design Dialogue to Yield Closure is another rule that we have followed however due to the lack of pages / features, it would be difficult to display conformance to this rule explicitly. The example for this rule in our application is that the application starts on the menu screen which then allows the user to navigate to the gamepage after selecting their desired game mode (singleplayer / multiplayer).

Prevent Errors can be observed when we attempt to navigate to the main menu page from the game page. This is to ensure that the user does not simply navigate to the main menu page without saving by accident.

Permit Easy Reversal of Actions cannot be observed for this application at this very moment but since our team has picked the Undo functionality, once this is implemented, it will allow for the player to revert their turn/ action in case of a misplay

Reduce short-term memory load can be demonstrated subtly in our game page's design. When playing the game physically, we would start off the game with 9 pieces in our hand and we are to place all 9 pieces on the board in order to move on to the next phase. However, for a software game, there are no tangible pieces for us to hold on to and subsequently there is nothing that allows us to keep track of the pieces that we have not already placed other than memorising. To overcome this, we have annotations on the top of the screens (as well as images of the pieces) that indicate how many pieces that are left to be placed on the board. This removes the need for the player to remember information like how many pieces there are left to be placed.

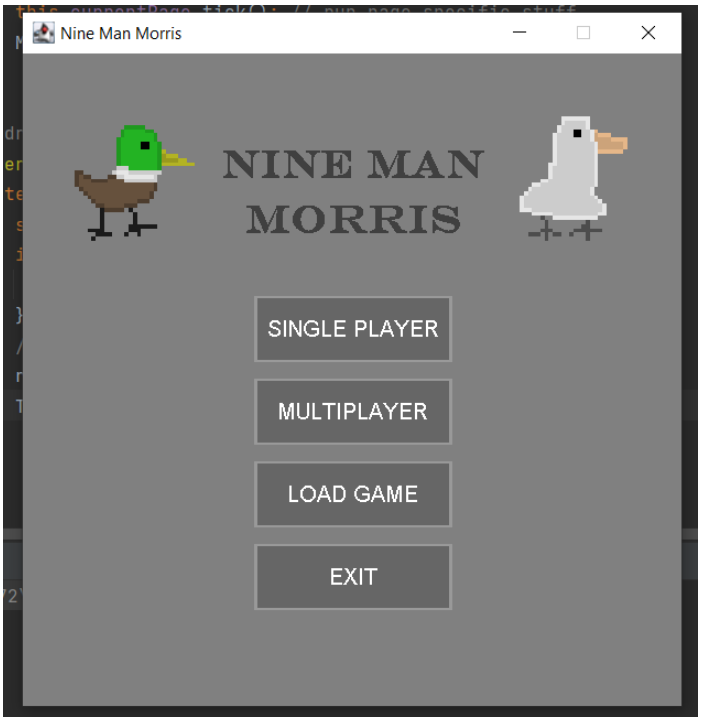


<3> Portability

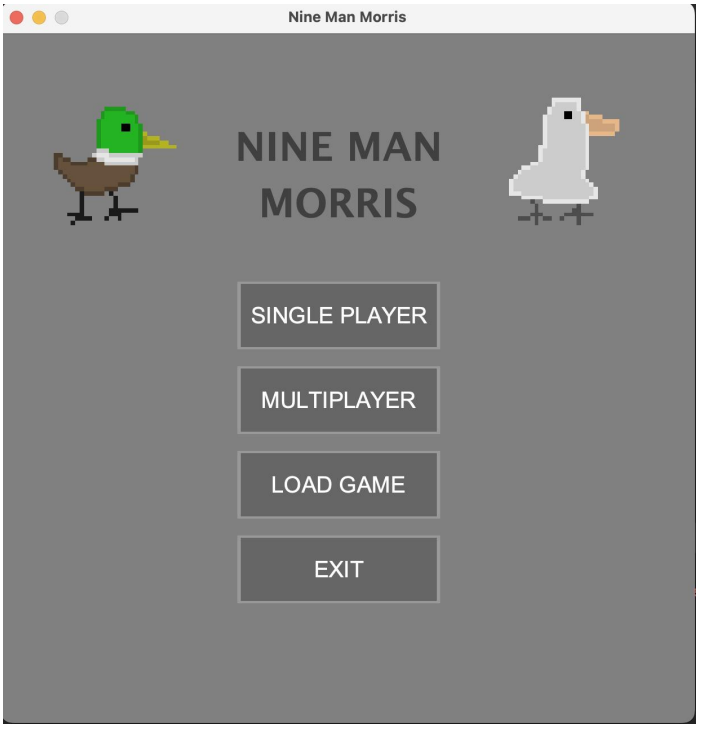
The game is exported to a .jar file, then converted to an executable (.exe) file as the .jar file acts as a midway point between Java and the executable file. After it is converted to an executable file, it can run on any operating system, mainly on Windows, Linux and MacOS. However, for Linux-based operating systems, running executable programs require an external application to run it. For example, in Ubuntu, Wine is required to be installed first in order to run executable files, but Wine is a free-to-use program which does not require any sort of payment, hence allowing the users to run executable programs for free on Ubuntu.

It could be further noted that Java is designed to be platform independent, meaning that jar files can run on any OS without modification to the program's code. As such, while the program is mainly intended to be exported as an exe, it is nonetheless possible to export the application as formats that are appropriate for other operating systems. As such, portability is preserved for the application.

Windows:



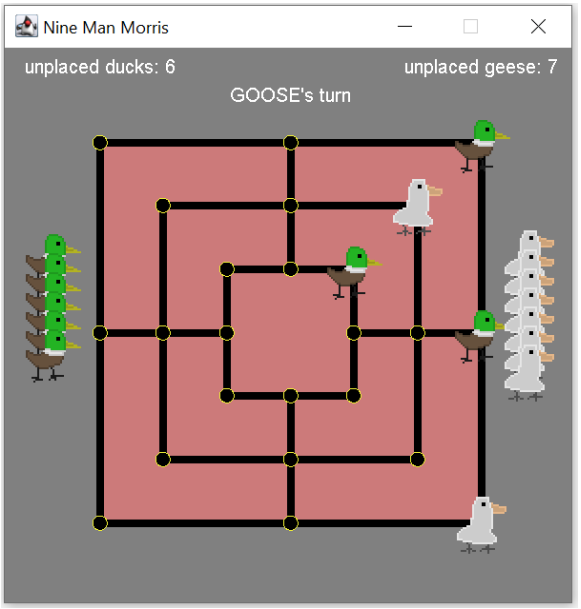
MacOS:



Human Values

Throughout the development of our Nine Men's Morris application, we decided to prioritize creativity as a human value. The main reason for this is that a multitude of implementations for Nine Men's Morris already exist, meaning that without a design that is unique and eye-catching, it would be difficult for our version of the game to stand out compared to other versions. This is especially true as the main reason for using the application is likely to be recreation and entertainment rather than for any sort of utility. It could further be noted that creativity enhances the overall user experience by stimulating a player's imagination and encouraging the exploration of new ideas through a user interface that is not merely usable, but also fun and entertaining to look at. As such, creativity is highly applicable for our application.

Multiple examples of creativity in the game can be seen from a screenshot of the game, as shown below:



While we initially considered a simpler concept design with each side's tokens being represented by white and black circles, we realized that such a design, while functional and understandable, would be excessively bland. In order to avoid this, we opted to represent the sides as animals instead of colours, with one side being represented as ducks and another side being represented as geese. By using avians for both teams, the application has a cohesive overall theme while still being unique and notable. We also took steps to avoid our focus on creativity becoming detrimental to usability by ensuring that ducks and geese look visually distinct, with ducks being rounder and more colourful.

Another measure we took to enhance creativity is adding duck and goose tokens on each side of the board in order to represent the number of tokens on each team that haven't been placed on the board. While a text with the corresponding number for each team is adequate to convey this information, we decided that a more visual representation would aid with creativity and would be more appealing to players.

Overall, multiple steps have been taken to take creativity into consideration in order to provide a better user experience for players.

Execution Instructions

- An exe file is provided to easily run the game on windows in the root folder of the repository

Video Demonstration

Below is the video link to the demonstration
https://youtu.be/8koUfTNg44c