Bilkent University

Department of Computer Engineering

CS 319 - Object-Oriented Software Engineering

Term Project - Analysis Report

Iteration 2

Project Name: Walls & Warriors

Group No: 2B

Group Name: OOPs

Group Members: Ali Babayev
Tunar Mahmudov
Merve Sağyatanlar
Çağla Sözen
Emin Bahadır Tülüce

Table of Contents

1. Introduction

For Object Oriented Software Engineering (CS 319) course, as Group 2B - OOPs, we've decided to design and implement a desktop application version of the game *Walls & Warriors*. A game based on "Troy" that challenges the cognitive skills, its first version, designed by Raf Peeters, was not widely played because of its complicated physical structure and redundant features **[1]**. For the second, widely successful version of the game, the design of the was simplified and its physical structure was improved to be more compact and more user-friendly. As OOPs, we've decided that for a game that faced such obstacles during its development, we could increase the functionality, user-friendliness and make it more entertainable by implementing it as a desktop application, as well as presenting a good design in terms of object oriented programming with its numerous distinct structures.

With already defined physical objects in the board game like Red Knights, Blue Knights, Walls and a High Tower this game, highly motivated us to come up with a good object oriented game design. In addition to the existing structure of the game we decided to implement additional features that will move with the natural flow of the game and utilize its object oriented design.

In our initial design plan, we are considering to include and add the following features:

- Campaign Mode
- Custom Mode
- Sound Options
- Knight Color Options
- How To Play Information
- Standard Challenges
- Wild Challenges
- Challenges with Increasing Difficulties
- Creating, Sharing, Playing Custom Challenges
- Displaying Hints

In this version of Walls&Warriors, structure of the game is based on dragging and dropping with mouse movements. The maps will vary according to the challenge type. In the two game modes: Campaign and Custom, user will have the chance to solve challenges provided with the game and challenges created by themselves or another player if acquire the Base64 encoding of that challenge respectively. These Custom challenges are created in the Challenge Editor which may be either a standard challenge or a wild challenge. This version also provides hints, changing the color of game objects, sounds, music and how to play information.

2. Overview

2.1. Game grid

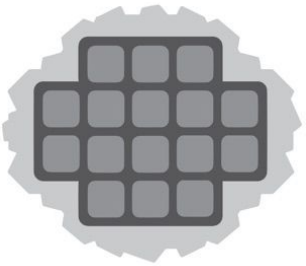


Figure 1: Game grid

The standard game grid consists of 14 blocks that are aligned as shown in Figure 1. However, the size and shape of the grid may vary in wild challenges (See Section 2.6 for all standard and wild differences). All of the other game elements will be placed on the grid. The knights and high towers stay on the blocks (square hollows) of the grid where the walls can be put on the borderlines of the blocks.

2.2. Knights

There are blue and red knights placed to particular blocks in each challenge. The player is not allowed to change their place or add/remove them from the grid. The knights are represented with circles that are colored accordingly.

2.3. High Tower

A high tower allocates two blocks on the grid and they act like two blue knights. The only difference is the two squares are connected by a solid rampart that does not allow the wall placements in between. (See Section 2.4 for all wall placement rules)

2.4. Walls

There are walls of different length and shapes to be placed by the player on the grid lines in each challenge. Some walls can have bastions on their specific places. The four standard walls are given in Figure 2. There are some rules apply when placing the walls on the grid, as listed below.

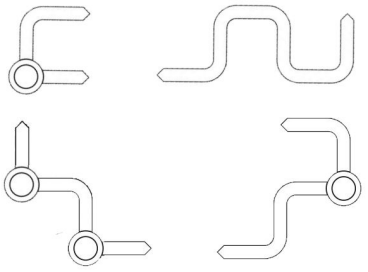
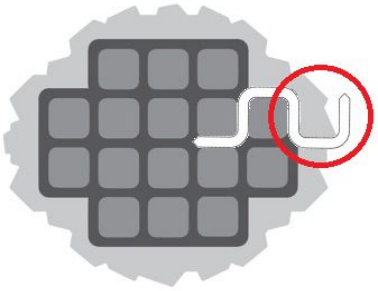
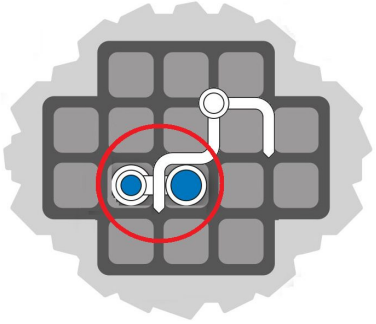


Figure 2: Standard walls



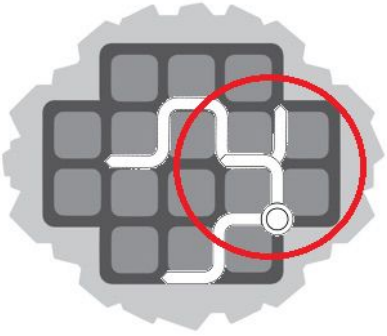
Rule 1
The walls cannot be placed outside of the grid.

Figure 3: Representation of rule 1



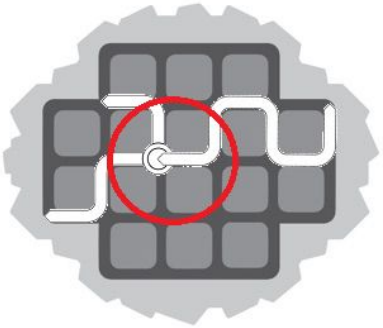
Rule 2
The walls can’t be placed where the high tower stands.

Figure 4: Representation of rule 2



Rule 3
The walls cannot be placed on top of each other.

Figure 5: Representation of rule 3



Rule 4
The walls cannot end on a bastion of another wall.

Figure 6: Representation of rule 4

2.5. Win Conditions

In order to win, the player should place all the walls to the grid in such a way that the following conditions are all satisfied.

- Blue knights and high towers should be surrounded by the placed walls from all sides.
- Red knights should be kept outside of the walls.

2.6. Challenges

In each challenge, a number of knights and high towers are given on the grid. The player is expected to place walls according to the rules to satisfy the win conditions.

2.6.1 Standard Challenges

The game grid is in a standard shape and size. In each challenge, the same 4 walls are given to the player (See Figure 2). The number of knights and high towers is limited by a number in standard challenges as listed below.

Maximum number of blue knights: 4
Maximum number of red knights: 3
Maximum number of high towers: 1

Other than these, all standard challenges have exactly one solution.

2.6.2. Wild Challenges

There are 10 different types of walls instead of 4. The creator of the challenge can specify which and how many walls to provide for the challenge he/she is creating. Moreover, the number of blue knights, red knights and high towers in the challenge are not limited. There might be more than one solution to wild challenges. The only limitation is that the challenge should be solvable by the given walls.

2.7. Game Modes

There are two different modes of the game.

2.7.1. Campaign Mode

In the campaign mode, the player is given a list standard challenges on a basis of increasing difficulties. Harder challenges are being unlocked as the player progresses on easier challenges. All the challenges in campaign mode are standard.

2.7.2. Custom Mode

In custom mode, the player can play challenges created by the game community. These challenges can be both standard and wild. There is no concept of a locked challenge for the custom mode.

2.8. Challenge Editor

All players are able to create their own challenges. In challenge editor, the player will pick the grid size and specify the number of red knights, blue knights and high towers. Afterwards, they can share their challenge with the game community to be played in the custom mode.

2.9. Settings

The user is able to modify the settings about the game including the volume of the sound effects and the visual theme pack.

3. Functional Requirements

3.1. Challenges

3.1.1. Campaign Challenges

The player will access this screen from the home screen using the “Campaign Challenges Button”. In this mode, the player is given a set of locked challenges that are default on the game. As the player succeeds in an easier challenge, harder ones will be unlocked. This way, the level of difficulty of challenges will increase gradually.

3.1.2. Custom Challenges

The player will access this screen from the home screen using the “Custom Challenges Button”. The player is given a set of challenges that are either created by himself/herself or the other players. Challenges created by other players can be imported using the code given for each challenge during challenge export. In this mode, all the challenges are unlocked and the player is free to choose any. The difficulty levels of challenges vary randomly within the challenges list.

3.2. Playing a Challenge

3.2.1. Resetting Progress

In the game screen, the player is provided with a “Reset Button” that enables the player to easily start over in the challenge. This feature is added to increase the user-friendliness of the game.

3.2.2. Helping the Player for a Challenge

In the game screen, if the player clicks on the “Check Button” without successfully completing the challenge, he/she will be provided with a visual and textual hint that shows and describes the missing piece or the fault.

3.3. Challenge Editor

The player will access the Challenge Editor from the home screen using the “Challenge Editor Button”. Challenge Editor allows the players to create their own challenges. There are two types of user-created challenges supported in the challenge editor, the player will be directed to each of the following screens according to his/her choice of challenge type using the buttons on the pop-up window.

3.3.1. Standard Challenge Editor

The player will access this screen from the pop-up screen on the home screen using the “Standard Challenges Button”. Standard Challenge Editor allows the player to select a desirable grid size within a range and use a limited number of game objects such as red/blue knights, walls and high towers. The player is expected to grab and drop the objects on the game grid appropriately according to the rules.

3.3.2. Wild Challenge Editor

The player will access this screen from the pop-up screen on the home screen using the “Wild Challenges Button”. Wild Challenge Editor allows the player to select a desirable grid size within a range and provides a non-limited number of game objects. The player is expected to grab and drop the objects on the game grid appropriately according to the game rules.

Once the player chooses the challenge and decides upon the number of objects and locates them on the grid, he will be asked to write his/her name and an optional short description of the challenge. The challenge will be accepted if there exists at least one solution, provided by the creator himself/herself by also locating the walls pieces on the grid appropriately. Also, if the challenge does not satisfy the constraints of building a challenge (i.e. There must be at least one knight from each color). If so, a copyable challenge code will be generated and given to the player in order to share this challenge with the other community members. The creator of the challenge will also have the option to add the level directly to his/her custom challenges using the button “Add as level”.

3.4. How To Play

The player will access this screen from the home screen using the “How To Play Button”. This option presents the rules and tips of the game. Basic information about the following is provided in this screen on different tabs,

Rules: Wall placement rules, win conditions, definitions etc.
Game modes: Definitions of the game modes and their differences
Challenge editor: Description for usage of the challenge editor
Controls: Description of the input controls

3.5. Settings
The player will access this screen from the home screen using the “Settings Button”. To make it user-friendly, game settings can be adjusted by the player. On this screen, the player is free to change the following features of the game: music volume, sound volume, color pairs for the knights.

3.6. Credits
The player will access this screen from the home screen using the “Credits Button”. Developers of this game and their contact information are listed in this option in addition to the GitHub link of the game.

3.7. Additional Functional Requirements
3.7.1 Partially Solved Challenges
In some of the easy challenges, some walls will be positioned on the grid by default. This feature will provide ease for the beginner users and they will get accustomed to the game more easily. It will also increase the game’s playability by children, who are the primary target audience.

4. Non-Functional Requirements

4.1. Basic Non-Functional Requirements
4.1.1. User-Friendly Interface
Since Walls & Warriors is a game originally designed for children,so the user-friendliness of the game is highly significant.

- The actions to place a wall should be easy to perform. The player should not have to perform a pixel-perfect wall placement, instead the wall should align itself to the closest available location. The block length that makes the auto-alignment of the wall should be at least 100 pixels wide.
- The menus and challenge navigation should be easy to interact and the user should be able to create a challenge in maximum 2 minutes.

4.1.2. Performance
Performance is very important in Walls & Warriors because it is meant to be a simple and straightforward game. Thus, the following requirements should be satisfied:

- Frame Per Second (FPS) should not drop below 50 during the game.
- The program should not consume more than 400 MB of RAM.
- The executable file should not take more than 200 KB of HDD storage.

4.1.3. Reliability
The player should not be able to import or export any unsolvable challenge. Thus, there should be an algorithm to check the challenge solvability before each export or

import action on the exported/imported challenge. Our aim is to sure that all challenges are solvable.

4.2. Additional Non-Functional Requirements

4.2.1. Extendibility

We want Walls & Warriors to be extendible since the game structure is open for features and rules.

- The algorithms that run in the game should be able to run with any type of wall shape, with any number of grid components and with any grid size. This way, the new grid components and different sized grids can be easily added to the game.
    - Requirement: The solve-checking method should accept a parameter grid that is not in the standard shape and should give the correct output.
- The grid components should be extendible as well. A new type of wall should be able to be added to the game easily without changing the other entities.
    - Requirement: Creating a new wall type should not take more than 20 lines of code.

5. System Models

5.1. Use-Case Diagram

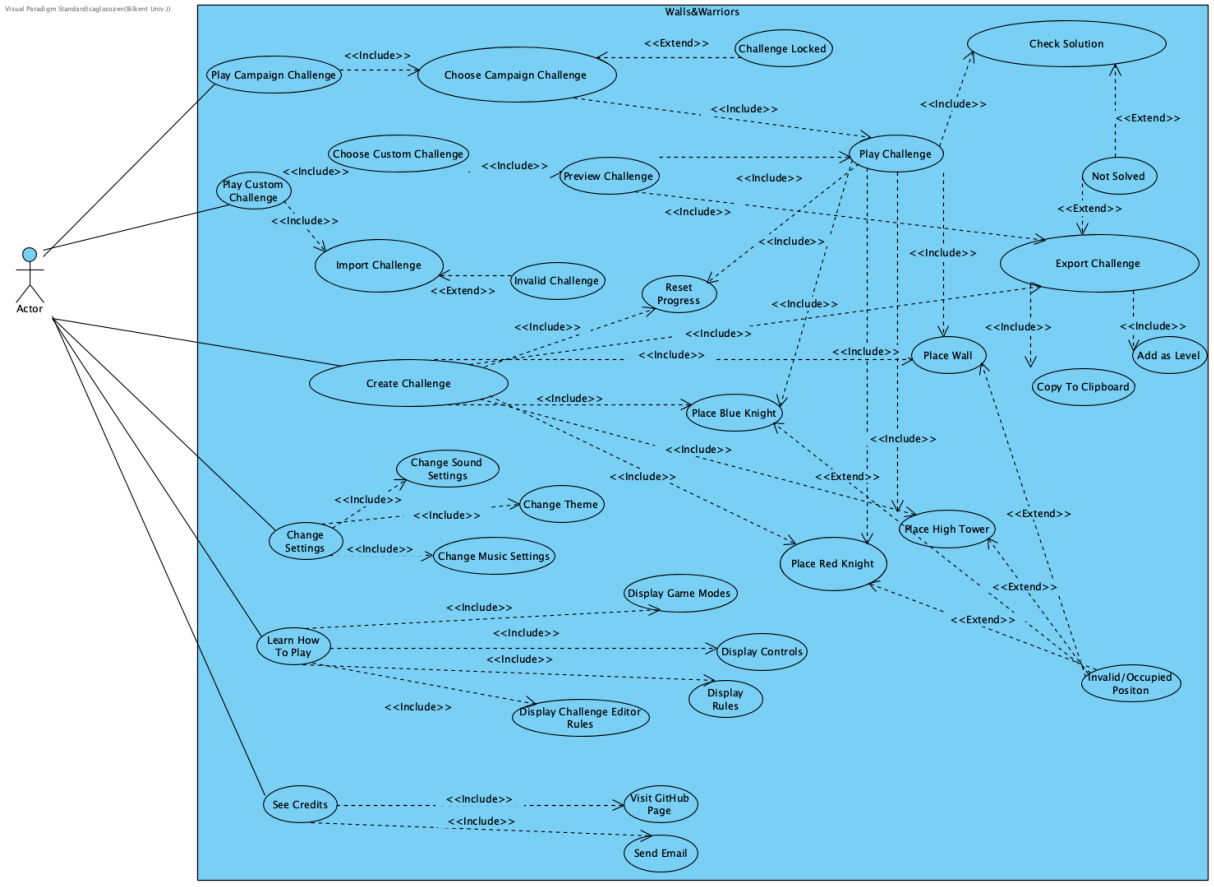


Figure 7: Use Case diagram

Use Case #1

1. ***Use Case***: Play Campaign Challenge
2. ***Participating actor***: Player
3. ***Entry Condition***: Player should be in the main menu
4. ***Exit Condition***: Player accomplishes the challenge
5. ***Success Scenario Event Flows***:
    1. Player chooses to enter Campaign Challenge menu
    2. Player chooses one of the unlocked maps
    3. Player chooses one of the given wall pieces
    4. Player places it into the map
    5. Player repeats the two previous process until placing all of the given walls correctly

6. Game displays success message
7. Game returns to choosing the map menu
8. Game unlocks the next locked map

6. *Alternative Event Flows*:

1. If player places the walls incorrectly
    a. Game gives chance to check mistakes
    b. Game displays the mistaken place
2. If player wants to return main menu
    a. The system asks for confirmation to exit
    b. Player confirm the choice
    c. The system leaves the game and returns to main menu

Use Case #2

***1. Use Case*:** Play Custom Challenge

***2. Participating actor*:** Player

***3. Entry Condition*:** Player should be in the main menu

***4. Exit Condition*:** Player accomplishes the challenge

***5. Success Scenario Event Flows*:**

1. Player chooses to enter Custom Challenge menu
2. Player chooses one of the challenges
3. Player chooses one of the given walls
4. Player places it into the map
5. Player repeats the two previous steps until placing all of the given walls correctly
6. Game displays “success” message
7. Game returns to the “choosing a map” menu

***6. Alternative Event Flows*:**

1. If player places the walls incorrectly
    a. Game gives chance to check mistakes
    b. Game displays the mistaken place
2. If player wants to return main menu
    a. The system asks for confirmation to exit

b. Player confirm the choice

c. The system leaves the game and returns to main menu

Use Case #3

***1. Use Case*:** Create Challenge

***2. Participating actor*:** Player

***3. Entry Conditions*:**

1. Player should be in the main menu
2. Player should choose to edit either “Standard” or “Wild” challenge

***4. Exit Condition*:** Player gets code for sharing his new challenge

***5. Success Scenario Event Flows*:**

1. Player chooses one of the challenge categories
2. Player determines the number of red and blue knights, and high towers
3. Player places them in any order into the map.
4. Player chooses a wall from the given available wall pieces
5. Player places the chosen wall into the map
6. Player repeats the previous 2 steps until completing the new map successfully
7. Player fills the description and author’s name part and chooses level of the new map.
8. Player exports the new created challenge
9. The system creates and displays a unique code to share

***6. Alternative Event Flows*:**

1. If player placed the walls incorrectly
    a. Game displays the “invalid” message
    b. Game returns to edit menu for player to correct mistakes
2. If player wants to return main menu
    a. System shows “Exit” menu
    b. Player selects “Exit”
    c. The system asks for confirmation
    d. Player confirm the choice

e. The system leaves the current menu and returns to main menu

3. If player wants to reset progress

a. Player chooses to reset progress

b. All figures on the map are removed

Use Case #4

1. ***Use Case*****:** Change Settings
2. ***Participating actor*****:** Player
3. ***Entry Condition*****:** Player should be in the main menu
4. ***Exit Condition*****:** The game applies the new setting changes
5. ***Success Scenario Event Flows*****:**

1. Player chooses to enter Change Settings menu
2. The system display settings
3. Player changes the level of the sounds
4. Player changes the level of the music
5. Player chooses knight colors
6. Player chooses to return main menu
7. The system asks confirmation for whether saving the changes
8. The system saves and returns to main menu

Use Case #5

1. ***Use Case*****:** How To Play
2. ***Participating actor*****:** Player
3. ***Entry Conditions*****:** Player should be in the main menu
4. ***Exit Condition*****:** The system returns to the main menu
5. ***Success Scenario Event Flows*****:**

1. Player chooses to enter How to play menu
2. The system displays rules
3. Player reads the tabs
4. Player chooses to return the main menu
5. The system asks for confirmation
6. Player confirm the choice

7. The system returns to main menu

Use Case #6

1. ***Use Case*:** Credits
2. ***Participating actor*:** Player
3. ***Entry Conditions*:** Player should be in the main menu
4. ***Exit Condition*:** The system returns to the main menu
5. ***Success Scenario Event Flows*:**
    1. Player chooses to enter Credits menu
    2. The system displays credits
    3. Player reads the given information
    4. Player chooses to return the main menu
    5. The system returns to main menu
6. ***Alternative Event Flows*:**
    1. If player chooses to visit our GitHub page
        a. The system opens the web browser
        b. The system visits to the our GitHub page's URL.

5.2. Dynamic Models

5.2.1. Activity Diagram

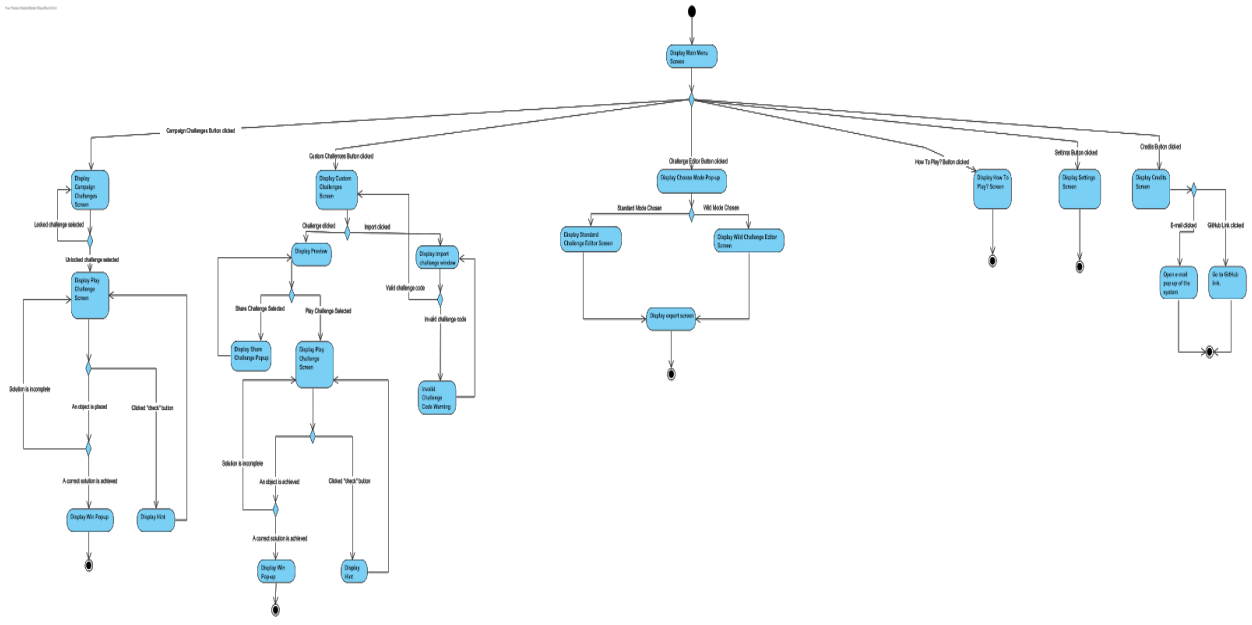


Figure 8: Activity diagram of the game

The main activities of the game are shown in Figure 8. In this diagram, the final nodes represent the ending of that activity In the main menu, the player is able to play campaign or custom challenges and create a level. These selections take the player to further screens like the level selection and challenge design. In each screen, proper warnings are displayed with invalid inputs to provide a reliable experience.Detailed versions of this diagram are shown in Figure 9, Figure 10.

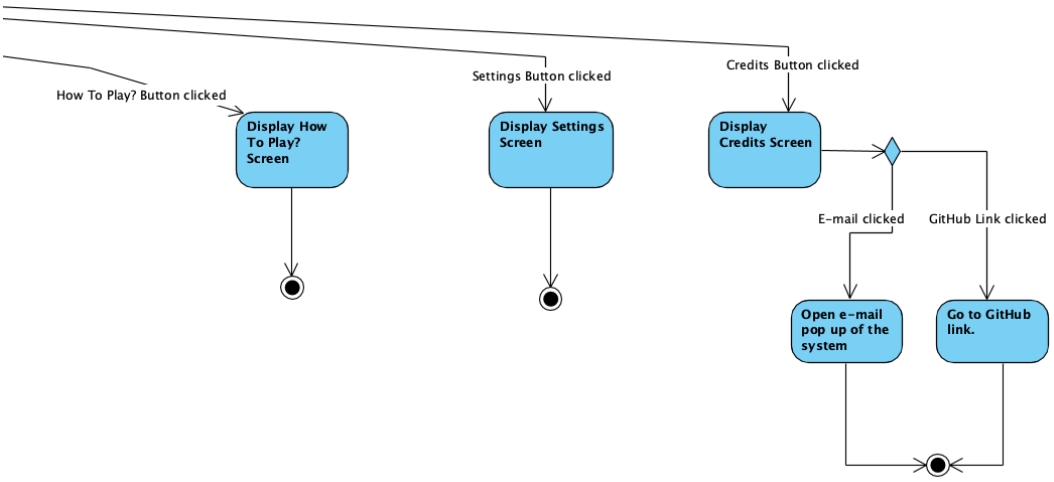


Figure 9: Activity diagram detailed part 1

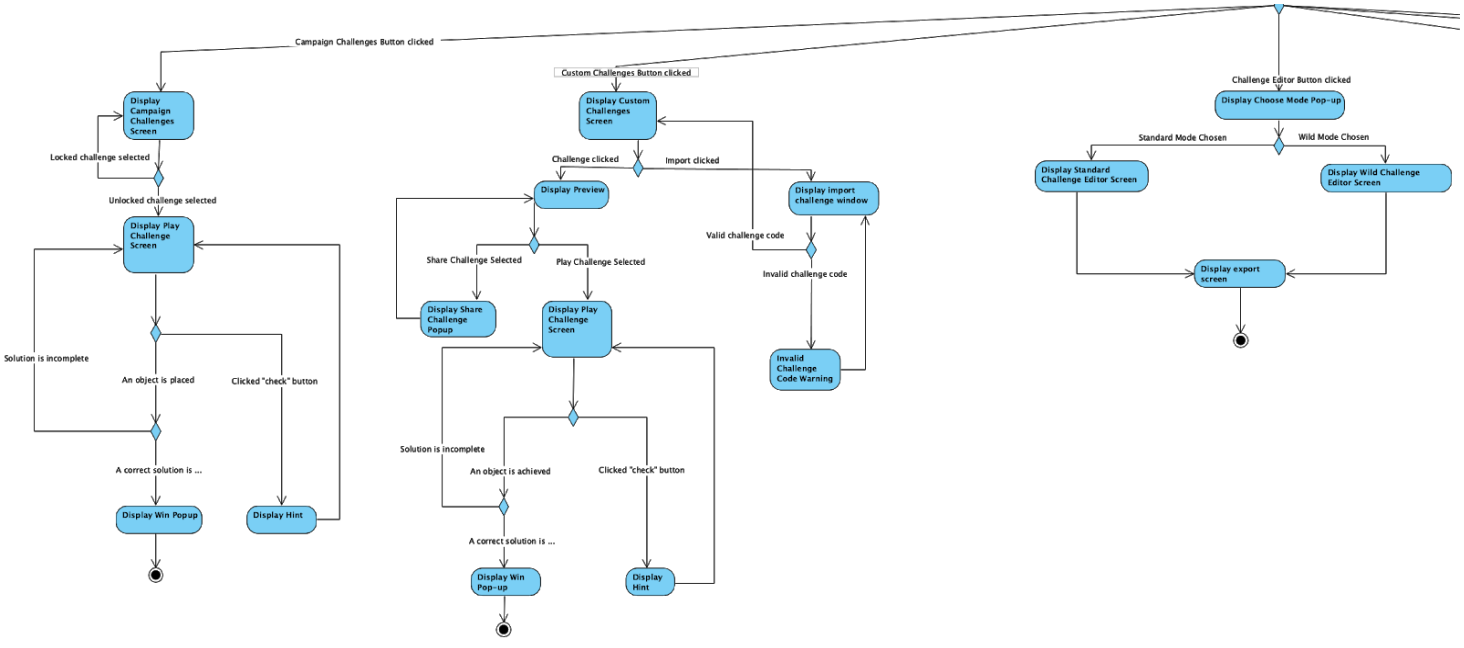


Figure 10: Activity diagram detailed part 2

5.2.2. Sequence Diagrams

NOTE: In the sequence diagrams, our objects do not reflect the implementation classes directly. For this reason, we choose to use names like “Mouse” and “Screen” instead of “MouseListener” and “Scene”, which made our analysis more closer to the real-life interactions about the game.

5.2.2.1. Start Playing a Campaign Challenge

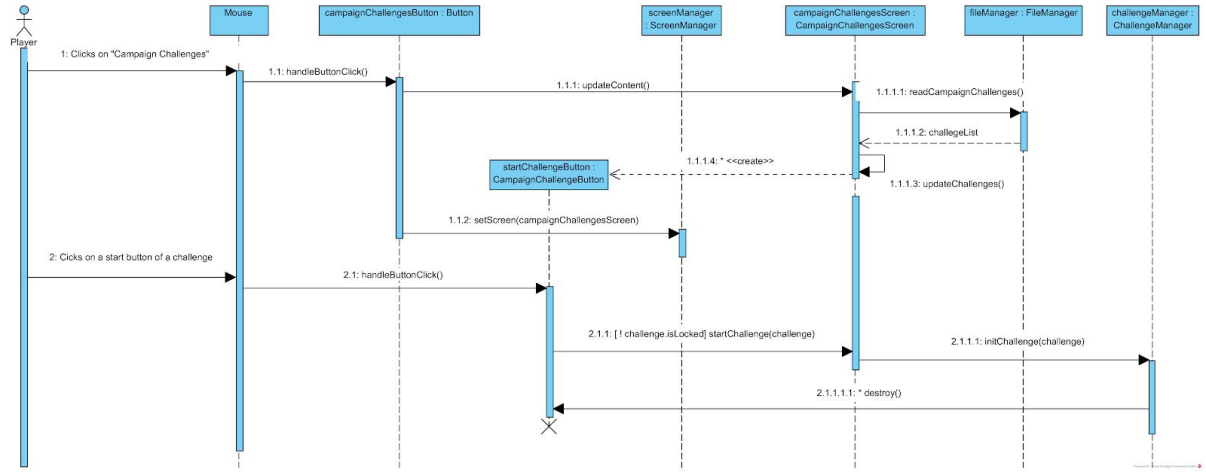


Figure 11: Playing “Campaign Challenge” sequence diagram

When the player wants to play a campaign challenge, he clicks on “Campaign Challenges”. The list of campaign challenges is displayed to the player. If the player clicks on a locked challenge, selection won’t open. If the player clicks on an unlocked challenge the game screen will be displayed with the chosen challenge’s configuration.

5.2.2.2. Start Playing a Custom Challenge

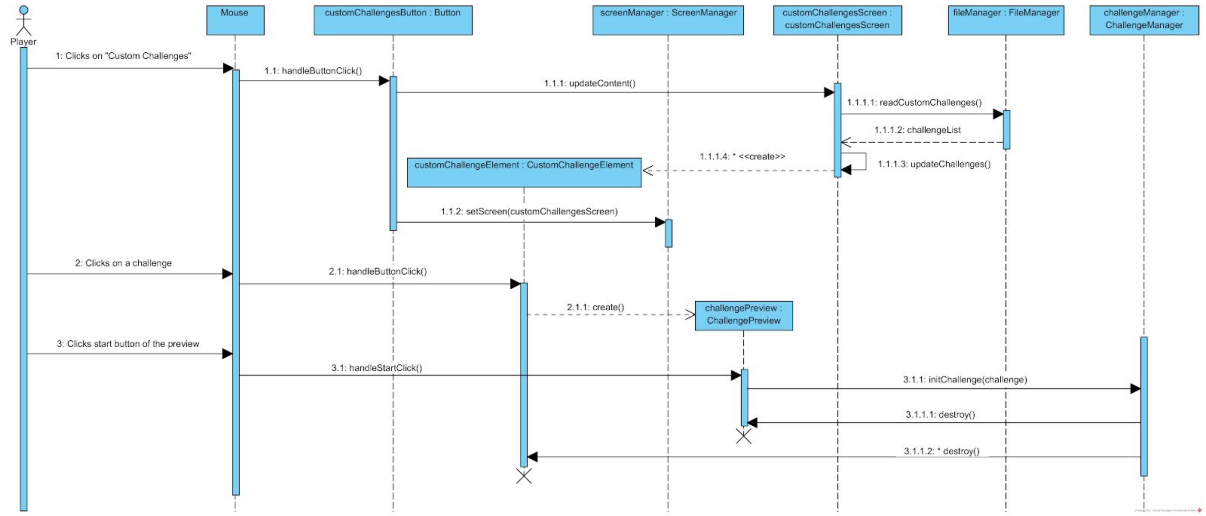


Figure 12: Playing “Custom Challenge” sequence diagram

When the player wants to play a custom challenge, he clicks on the “Custom Challenges”. The list of previously imported custom challenges is displayed to the player. If the player clicks on a challenge, the preview and information about the challenge will be shown. The player can click on “Play” to switch to the game screen with the selection, or there is also an option to get the challenge code to share the selected challenge.

5.2.2.3. Play a Challenge

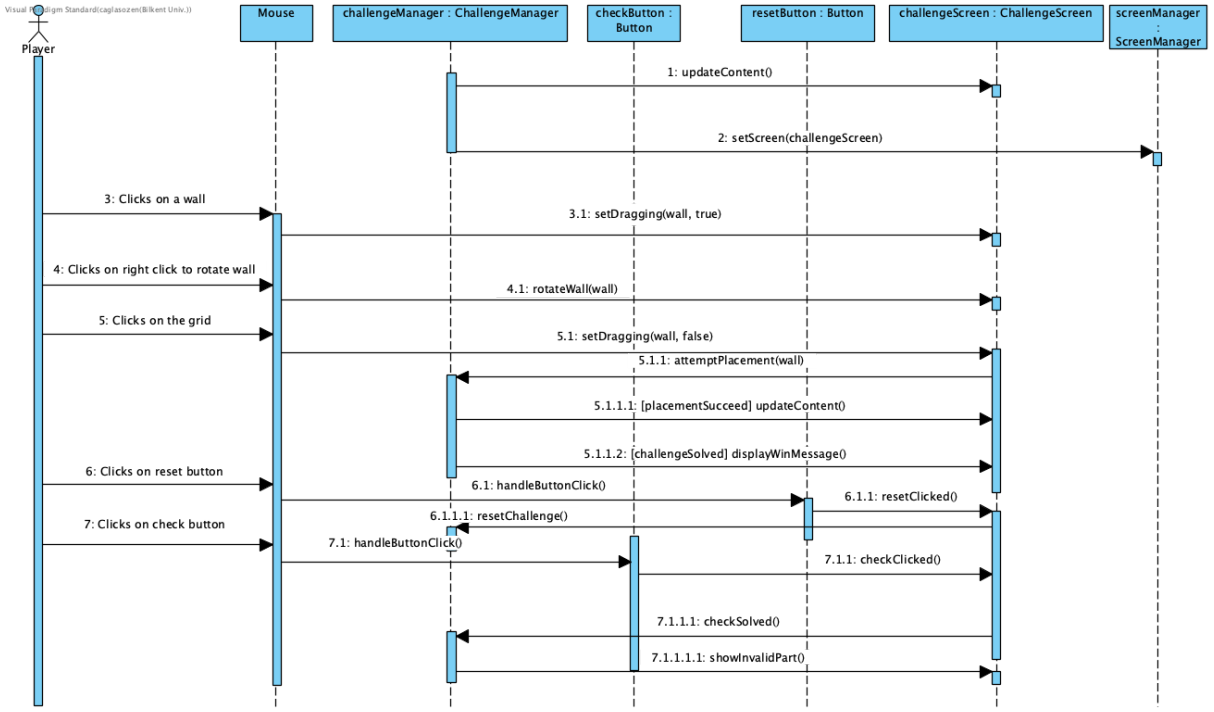


Figure 13: Play challenge sequence diagram

When the game screen is displayed, the user will see the configurations of placed elements (blue knights, red knights and high towers) along with the walls that are available for placing. The user drags a wall to the grid and the wall will be attempted to be placed to the nearest location. If the location satisfies wall placement rules (See Section 2.4) the placement will succeed, otherwise it will fail. After each placement the win conditions are checked and a winning message is displayed if they are satisfied. Also, the player can click on “Check” to see the non-satisfied win condition (i.e. invalid part).

5.2.2.4. Import a Custom Challenge

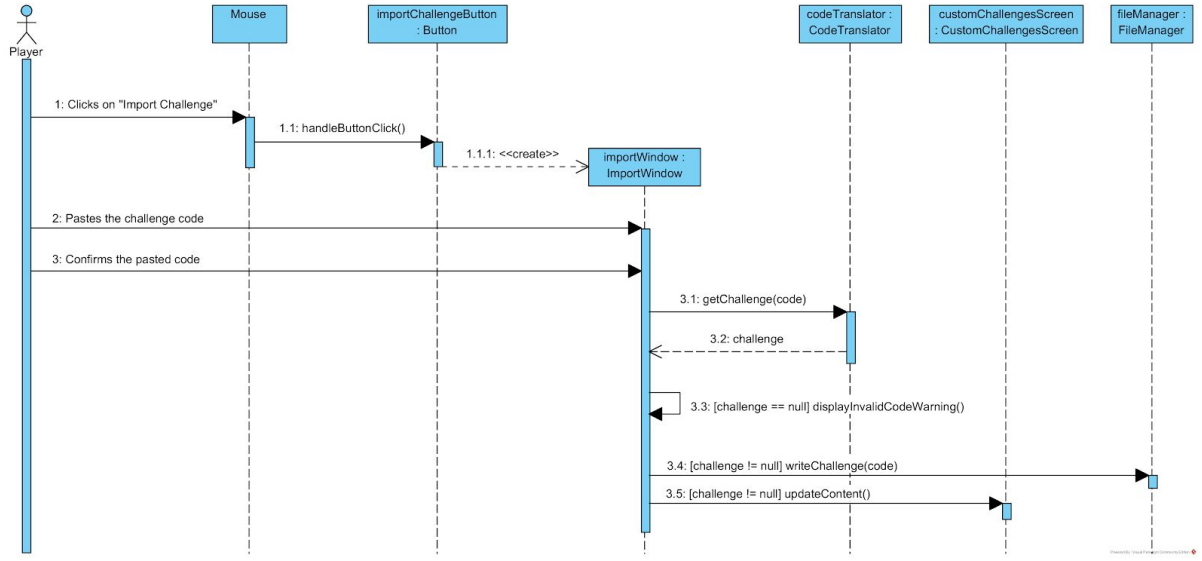


Figure 14: Importing challenge sequence diagram

The player has the code of a challenge in their clipboard. They click on the “Import Challenge” button and paste the code. The code is processed and analyzed. If it corresponds to a valid challenge, the challenge is added to the custom challenges list, otherwise, a warning will be shown to the player.

5.2.2.5. Create a Challenge

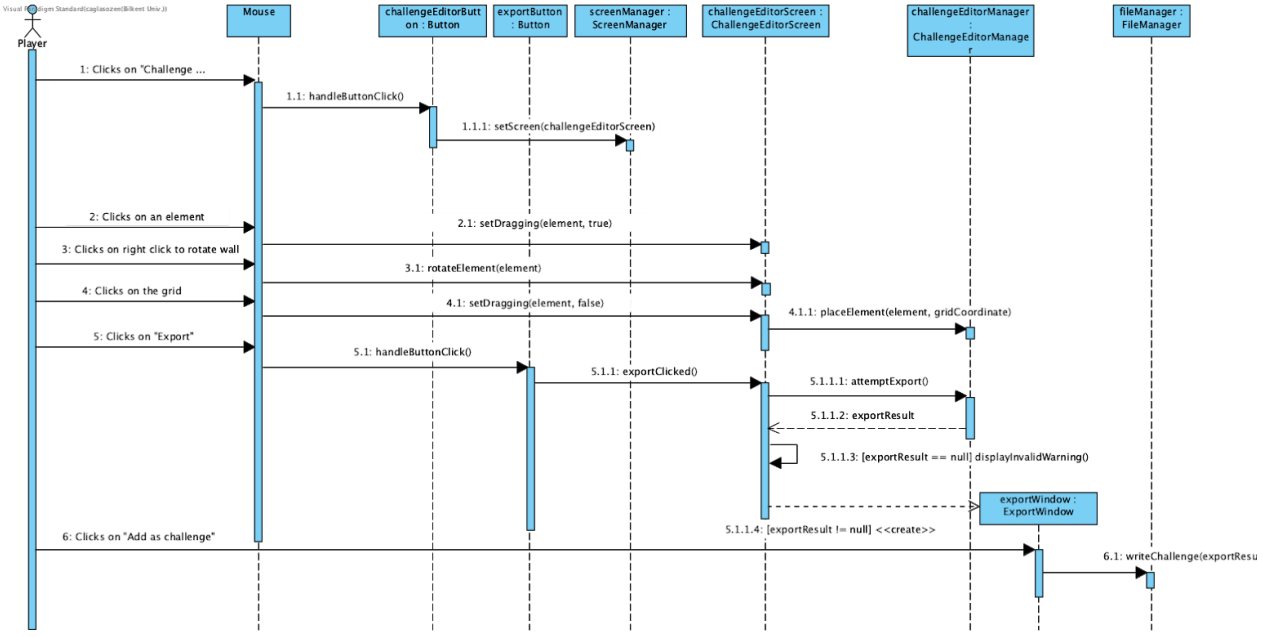


Figure 15: Creating challenge sequence diagram

When the player wants to create a new challenge, they click on the “Challenge Editor” button that is in the main menu. The elements (knights, high towers, walls) of the challenges will appear on the editor screen for the player to drag them to the grid. When the player is done creating his challenge, he clicks “Export” button and the code of the created challenge will be provided to the player. They can copy the code to share or they can directly add the challenge to their imported challenges list.

5.2.2.6. Check Player’s Solution

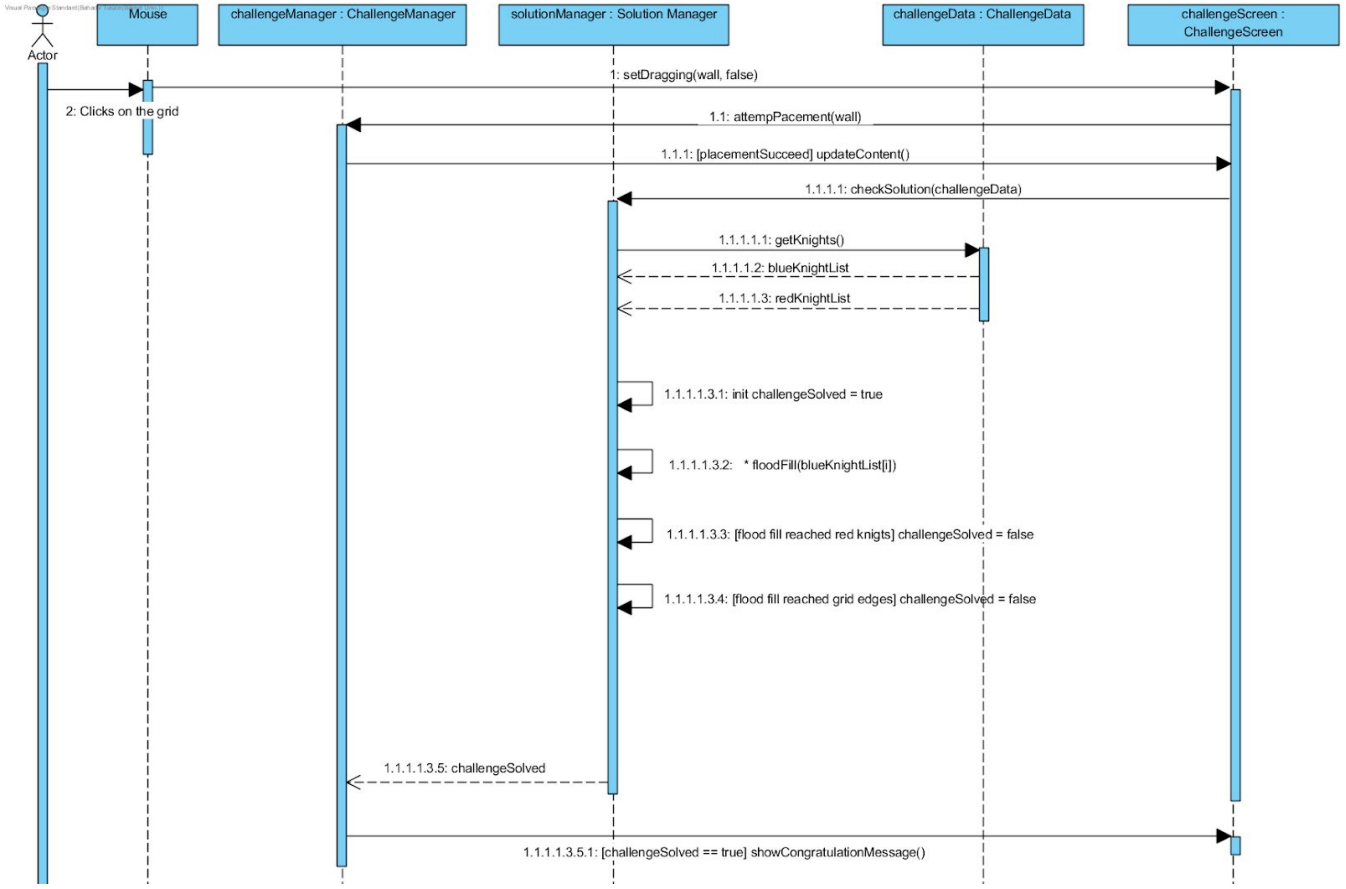


Figure 16: Checking solution sequence diagram

When the player clicks on the grid to place a wall, the game checks if the achieved state of the grid satisfies a valid solution to the current challenge. This process will be done in a function called checkSolution() by SolutionManager. The sketch of the algorithm can be seen in the sequence diagram that is in Figure 16 above. The basic procedure is as follows:

- Apply a “flood fill” algorithm on the grid blocks of blue knights.
    - If the grid blocks of red knights are filled with this algorithm, then red knights are not outside the walls.
    - If the edges of the grid are filled with this algorithm, then the walls do not form a closed structure.

After this procedure, the game will decide on whether or not to display the congratulations message to the player, according to the results.

5.2.3. State Diagrams

5.2.3.1. State of a Wall Piece

In the challenge screen, a wall piece can be in different states during the game. The states and the transitions of them are demonstrated in *Figure 17* along with their explanations below.

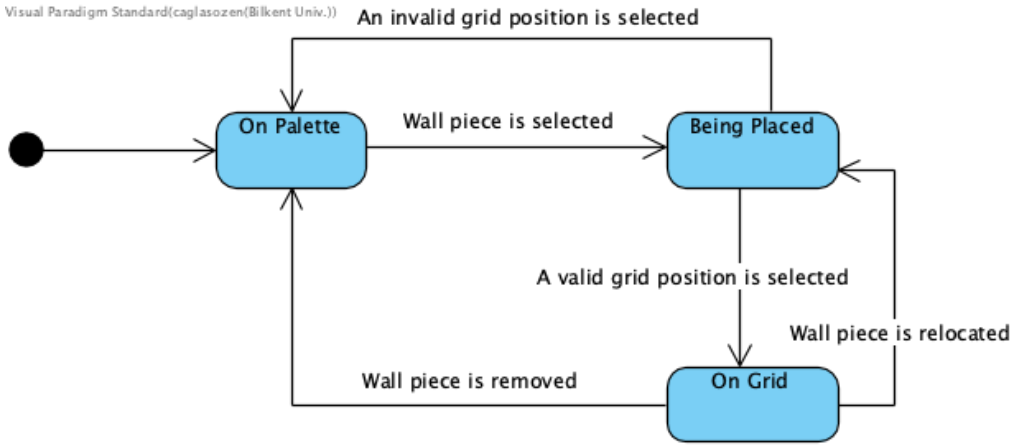


Figure 17: *State diagram for a wall piece*

On Pallette State: The wall piece is shown as a small preview and not occupying any space on the grid.
Being Placed State: The wall piece is selected and the player now can choose either a valid or invalid grid position.
On Grid State: The wall piece is placed to the grid and occupying some place on the grid. While in this state, it can be removed or relocated by the player.

- It should be noted that, the user interactions (the selections, removing and relocating process) which are defined in this process might differ with the future design choices. They could be controlled through mouse clicks, keyboard presses or screen touches. Here, we have only demonstrated the process as “selection” of wall pieces and grid positions.

5.2.3.2. State of the Challenge Screen

The challenge screen itself has a state of it own to decide on what to display to the player. This decision is made according to the states and their transitions given in the Figure 18 below.

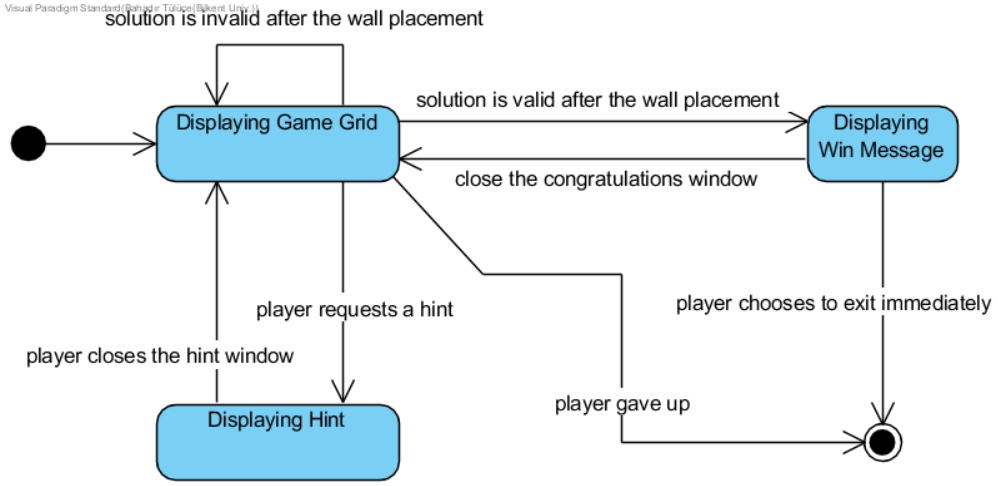


Figure 18: *State diagram for the Challenge Screen*

Displaying Game Grid State: The screen is displaying a game grid to the player so the player tries to solve the challenge while interacting with this grid.
Displaying Win Message State: The screen is displaying a “congratulations” message to the player. The player can immediately go back to the main menu or challenges screen, or they can choose to close this message window to take a look at their solution.
Displaying Hint State: The screen is displaying a hint to the player so that they can have an idea on the solution. They have to close this window to go on with the other interactions of the game.

5.3. Object and Class Model

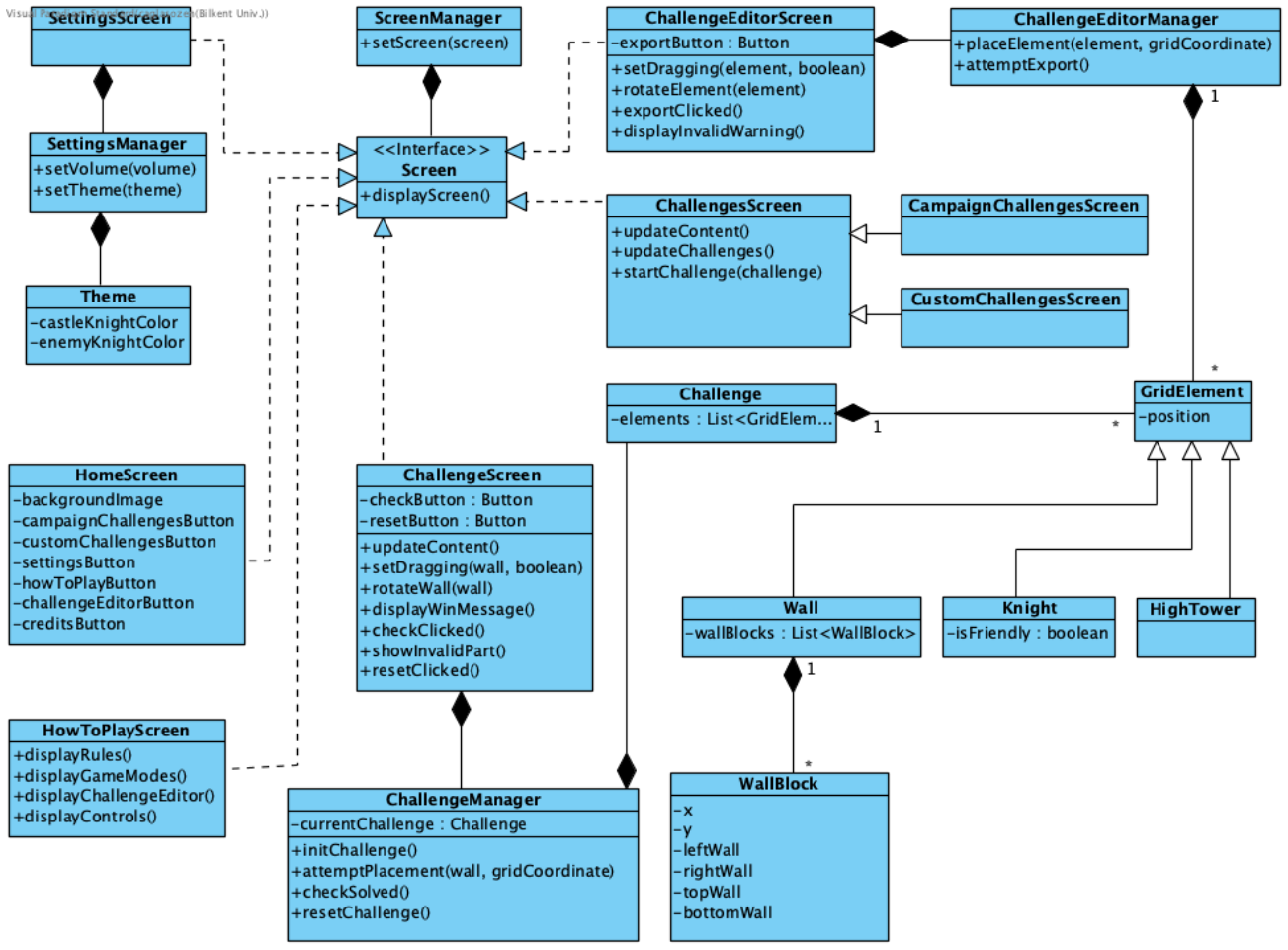


Figure 19: Class diagram for the game objects

5.4. User Interface

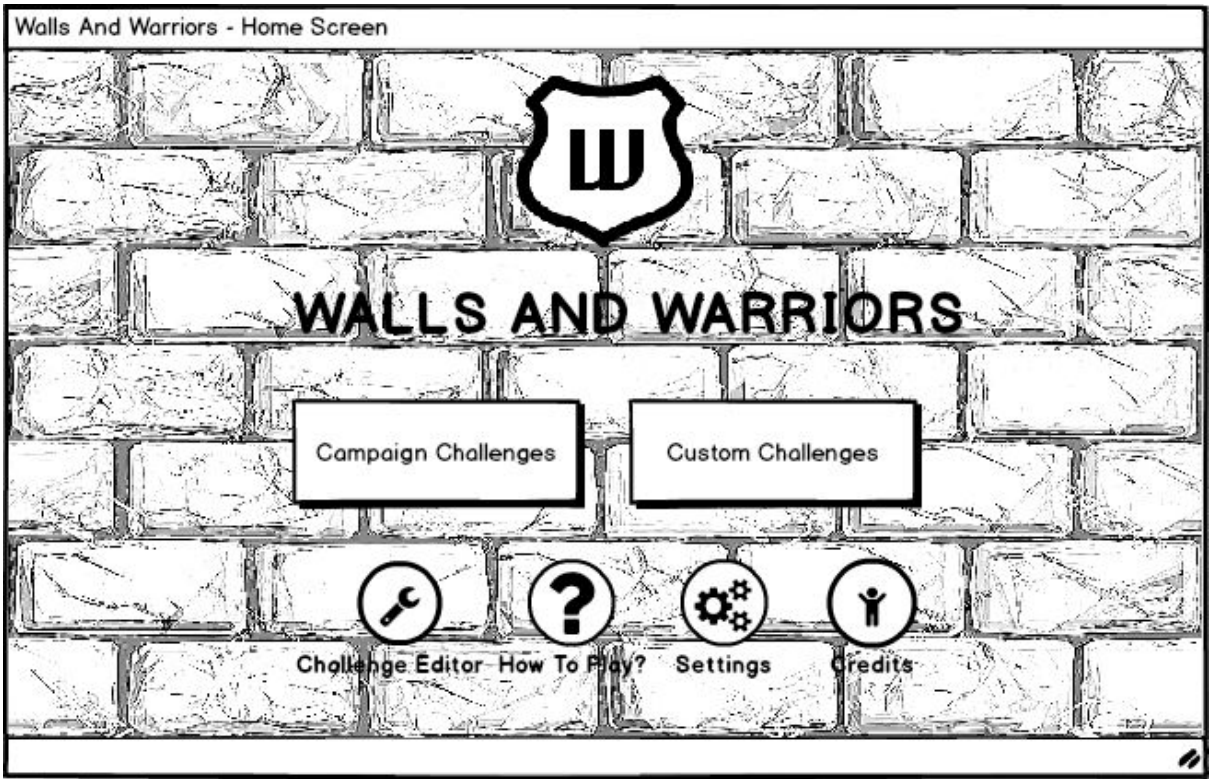


Figure 20: Home screen

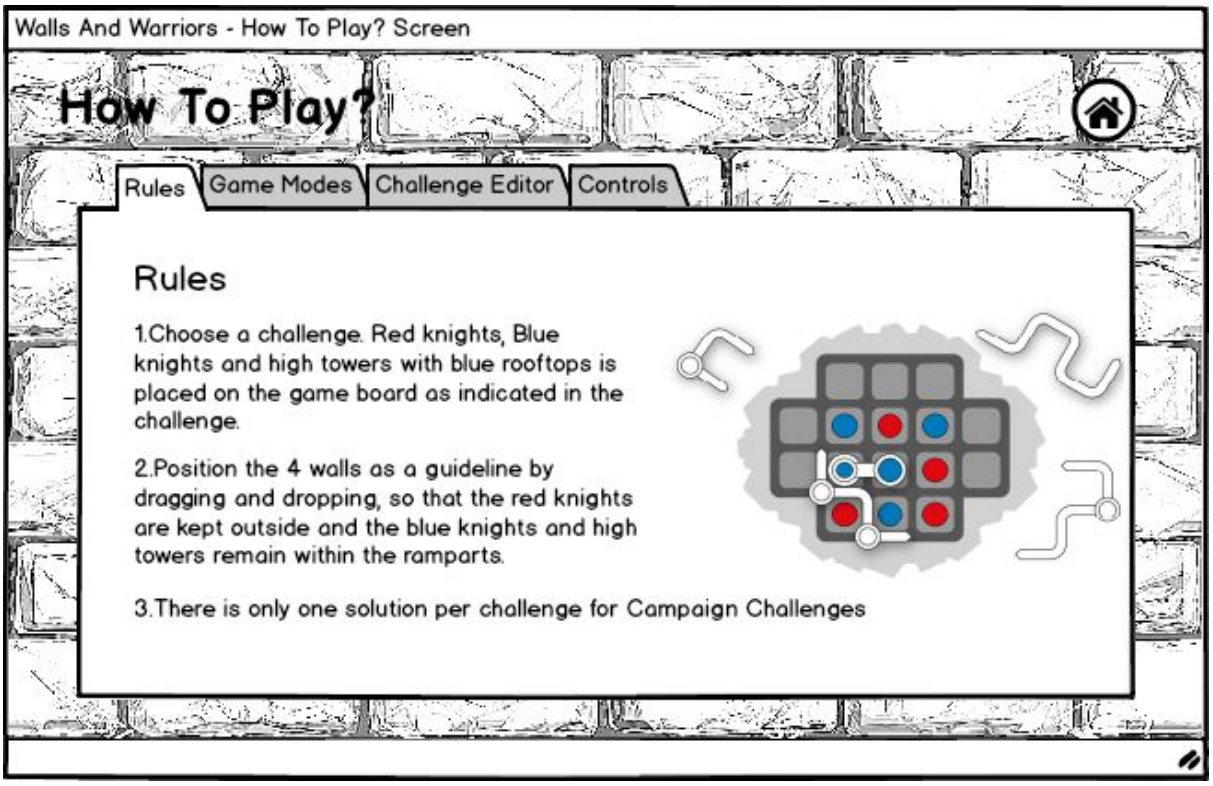


Figure 21: How to play - Rules

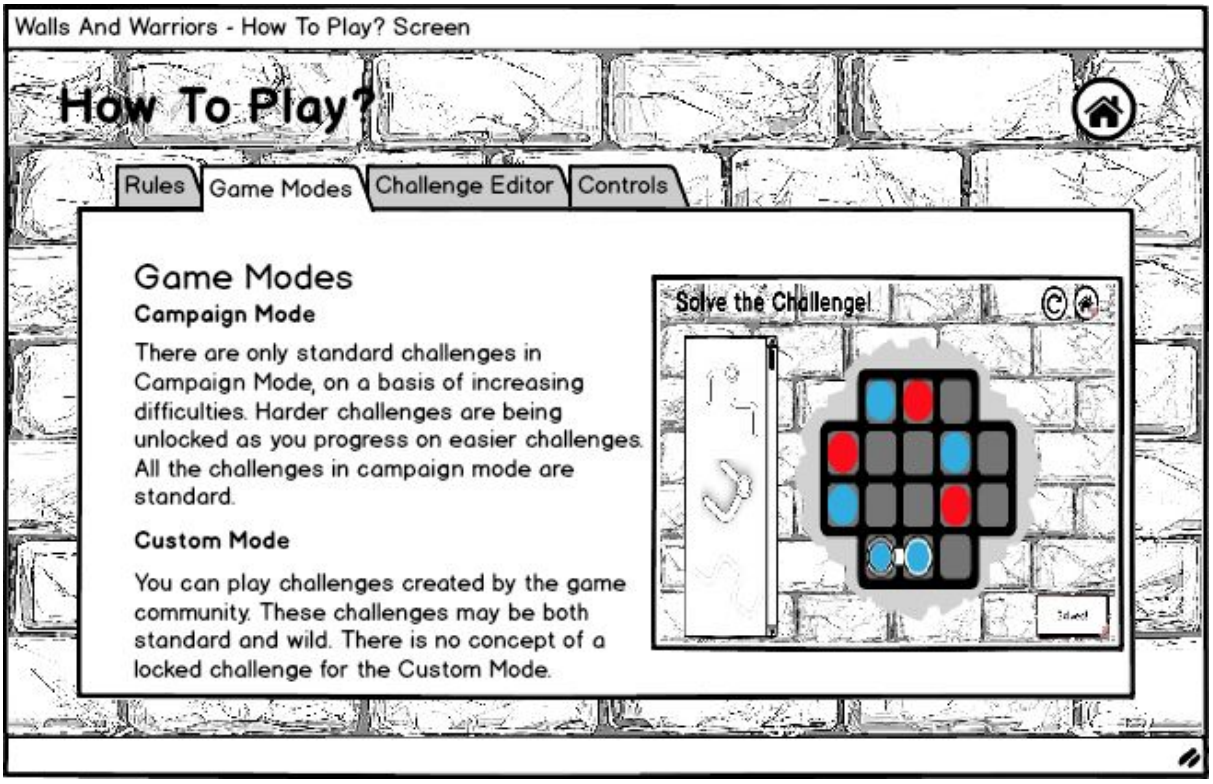


Figure 22: How to play - Game modes

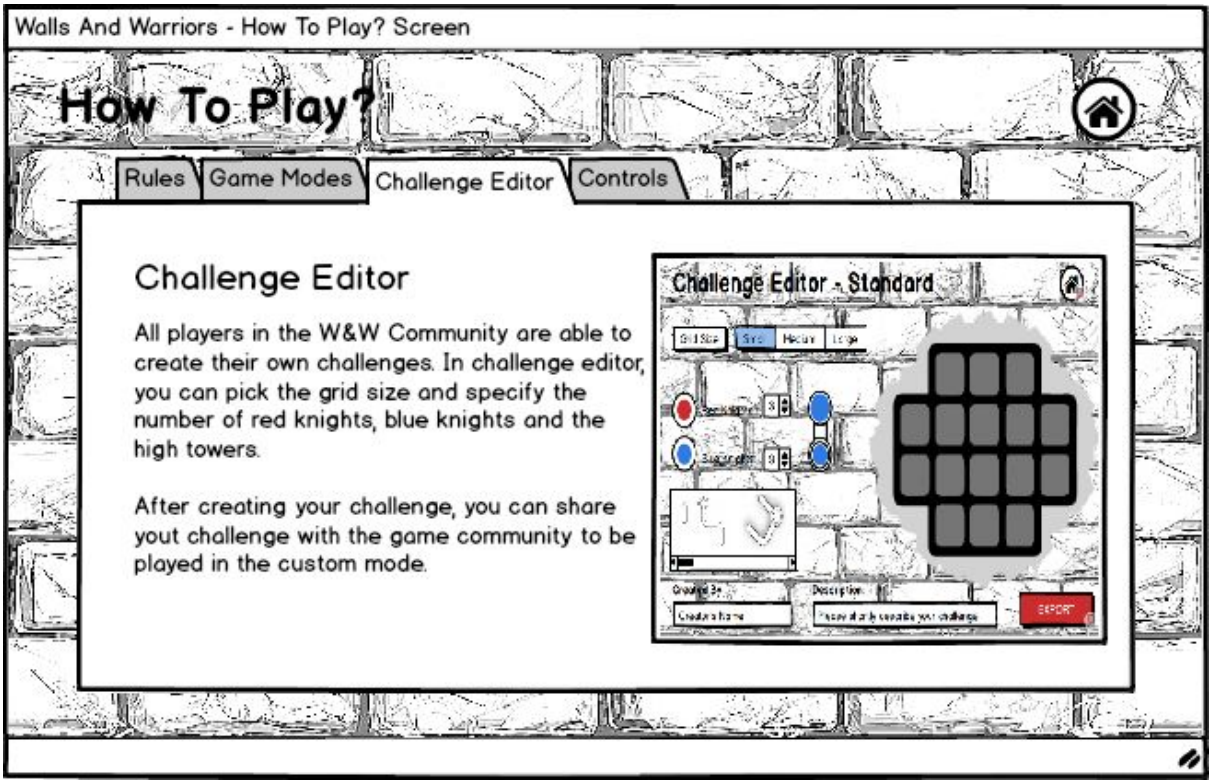


Figure 23: How to play - Challenge Editor

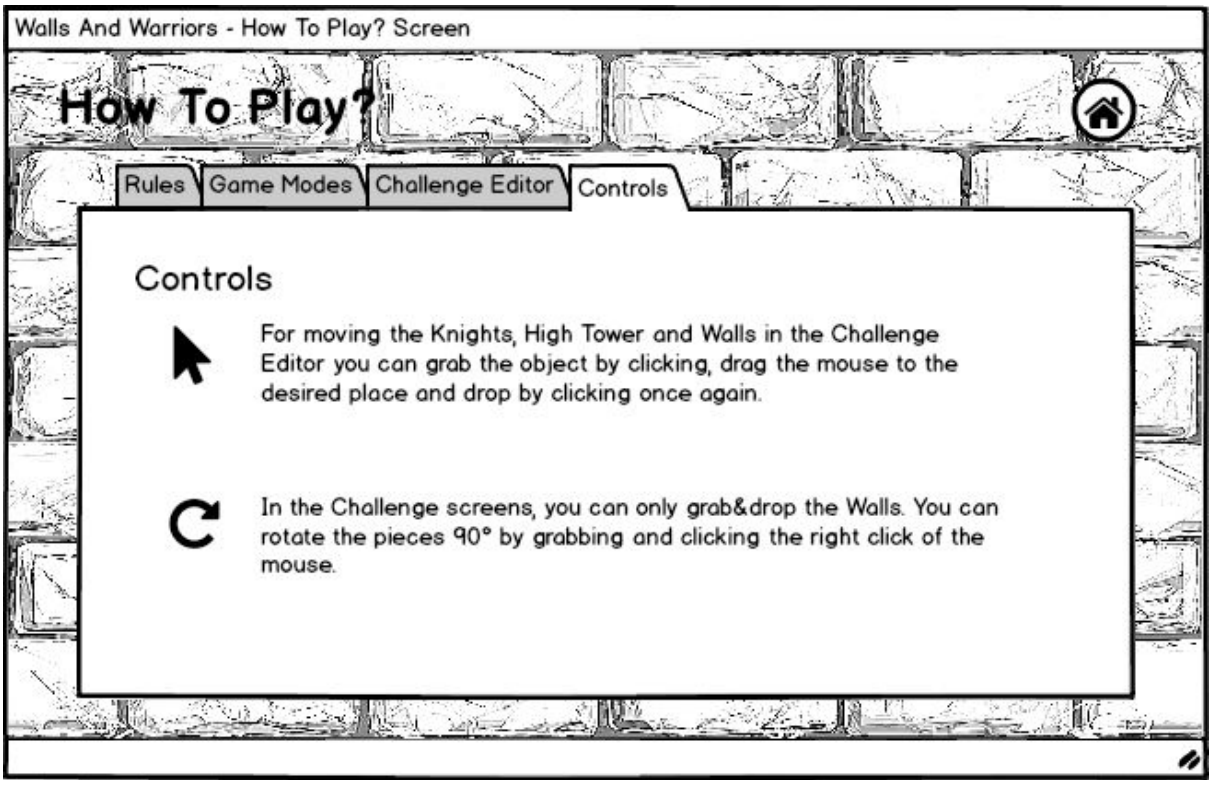


Figure 24: How to play - Controls

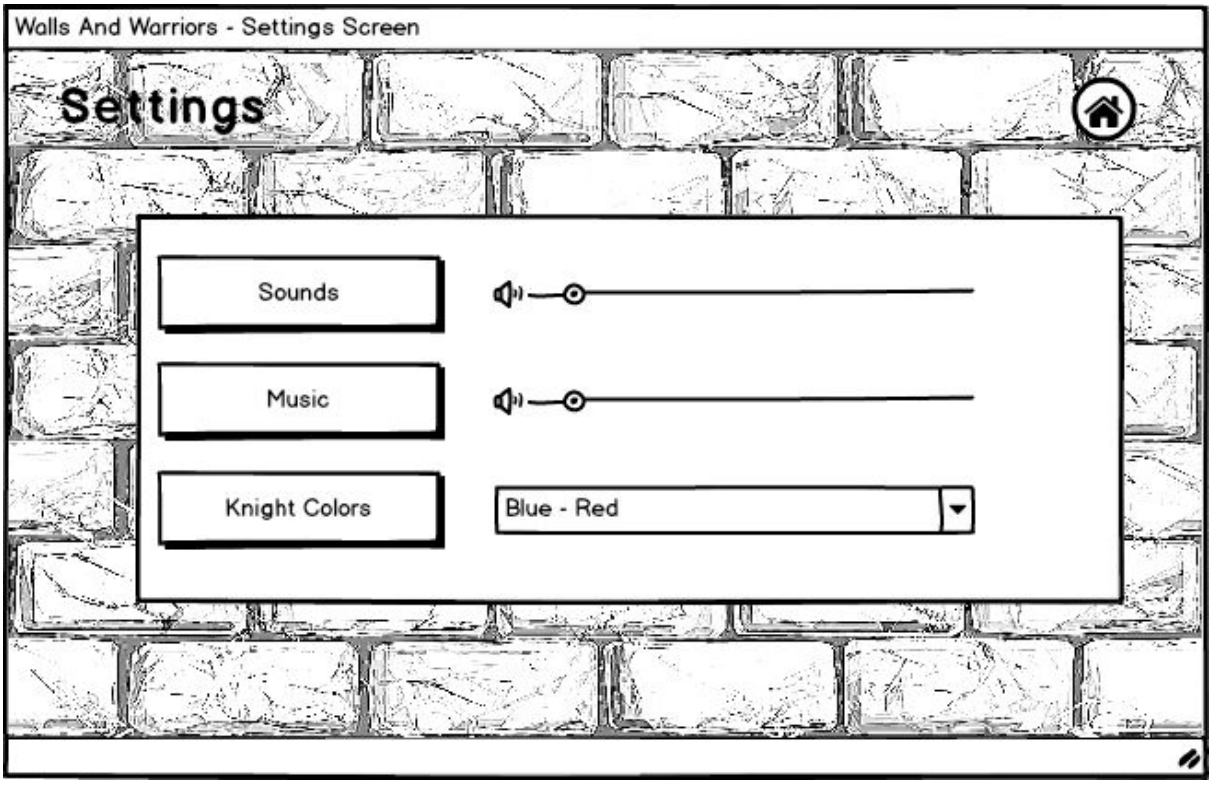


Figure 25: Settings

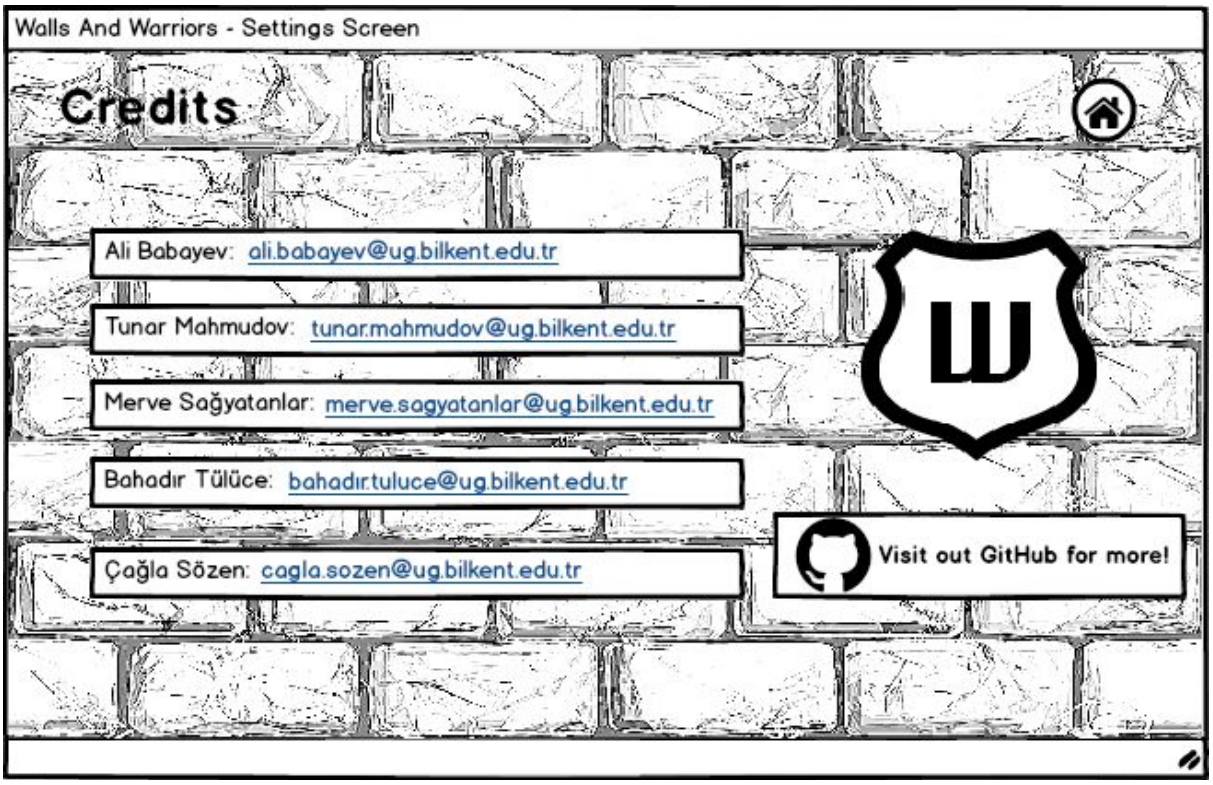


Figure 26: Credits

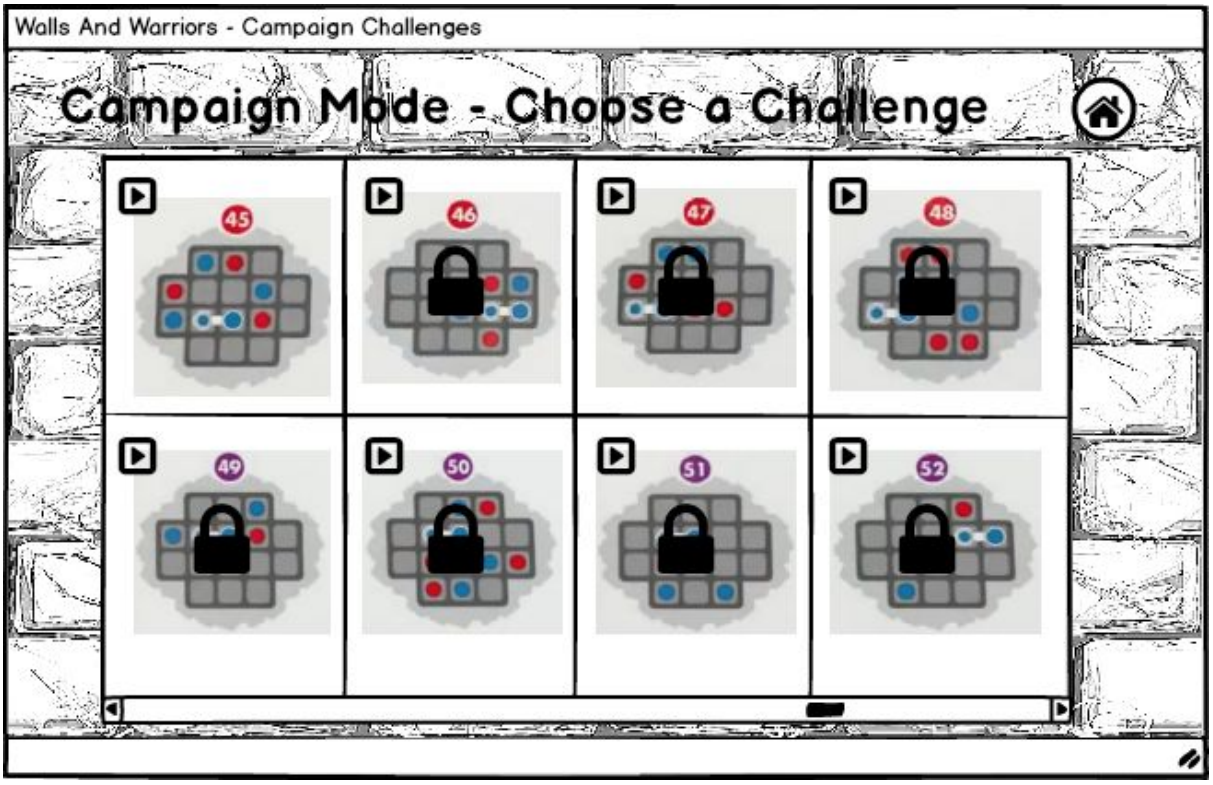


Figure 27: Campaign challenges

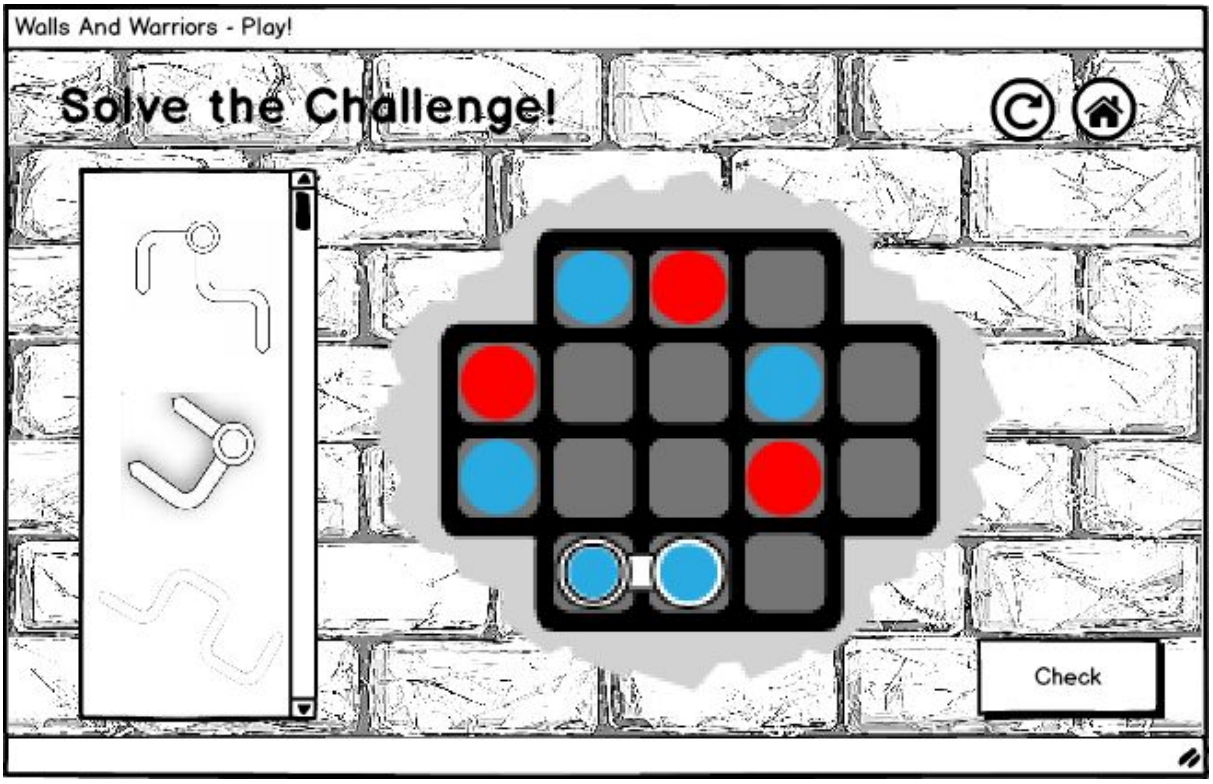


Figure 28: Game screen

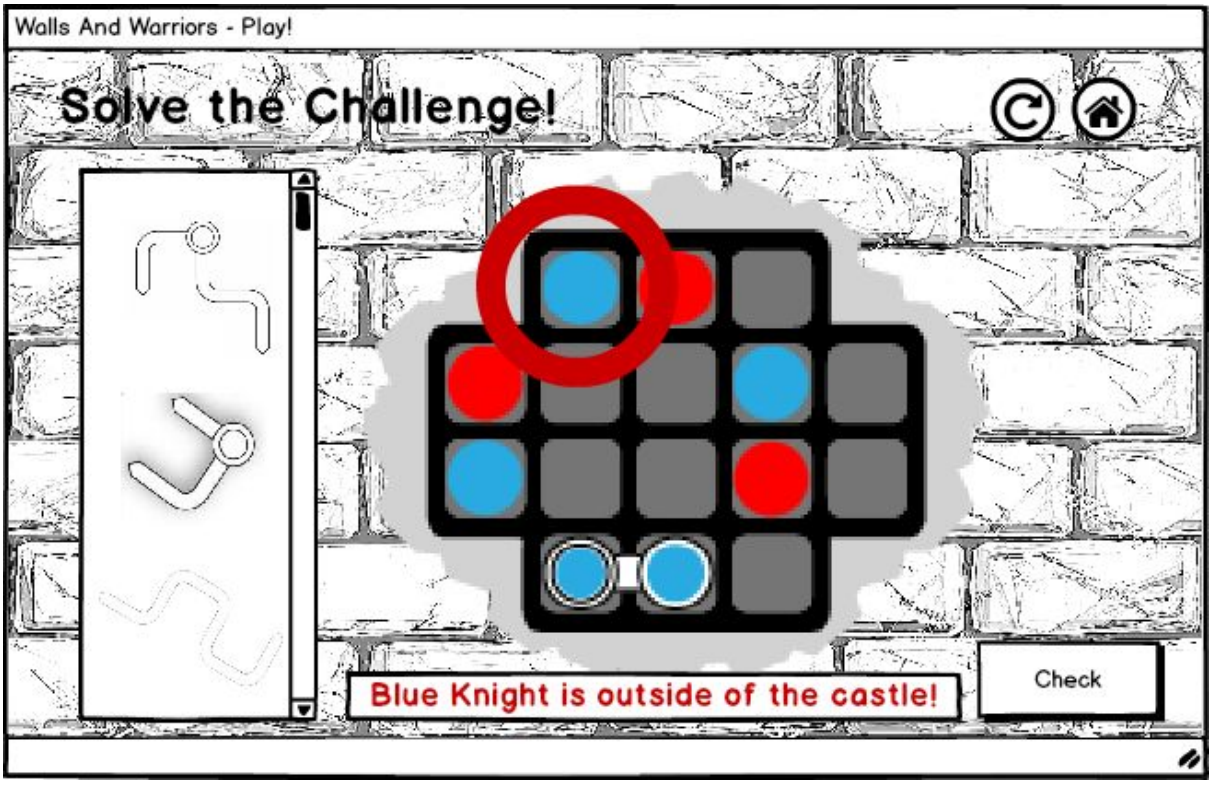


Figure 29: Game screen's warning message after clicking Check button

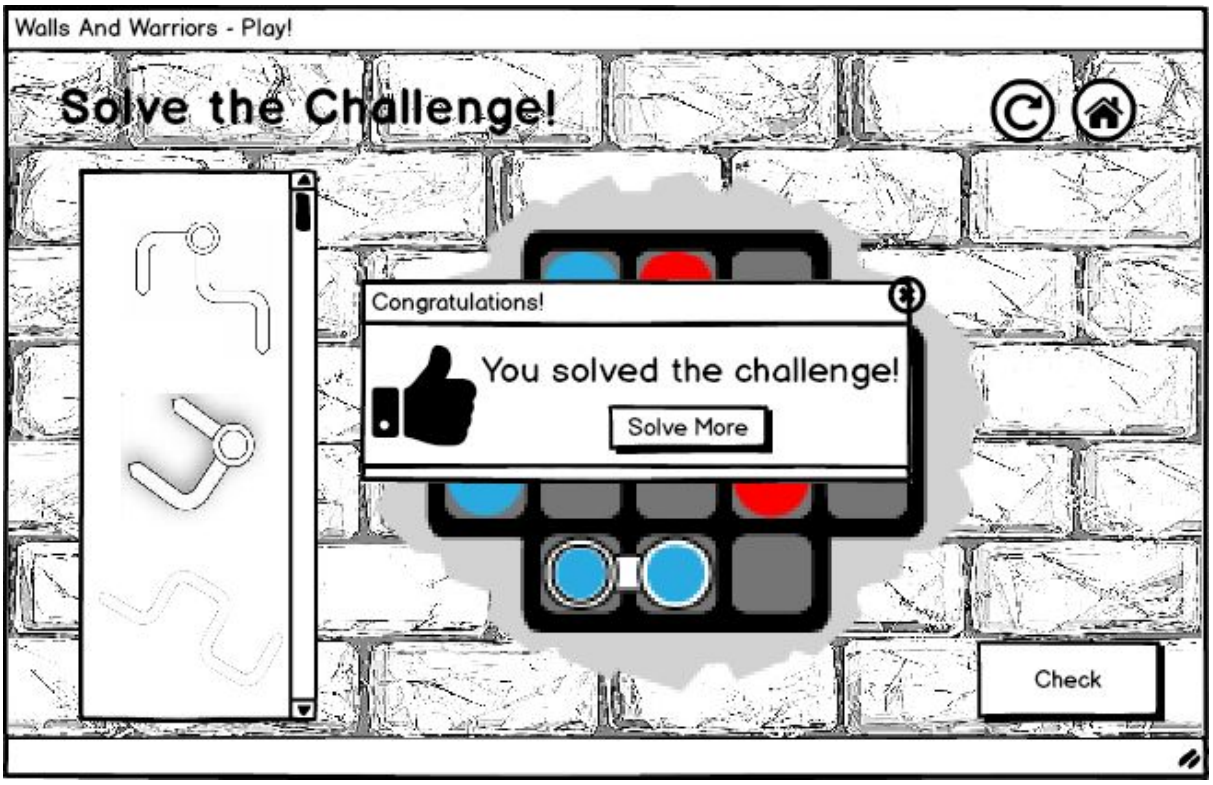


Figure 30: Win screen

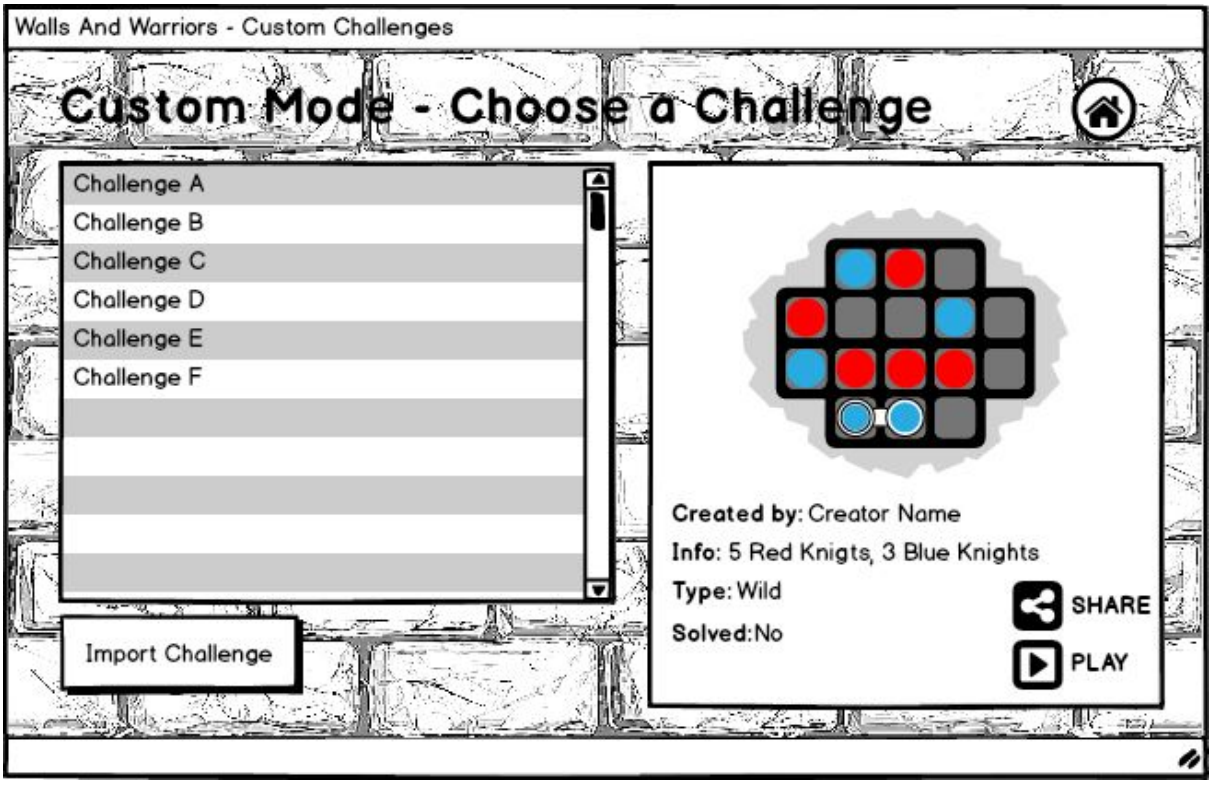


Figure 31: Custom challenges

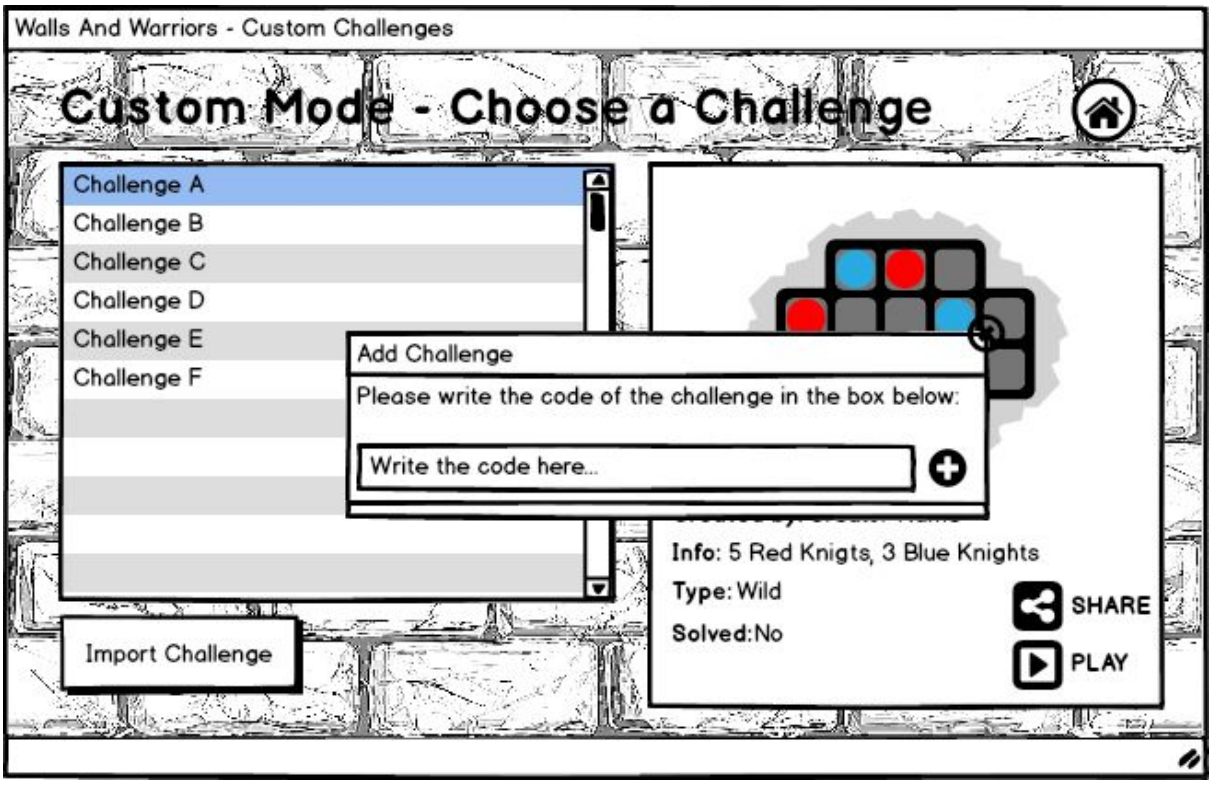


Figure 32: Import a challenge

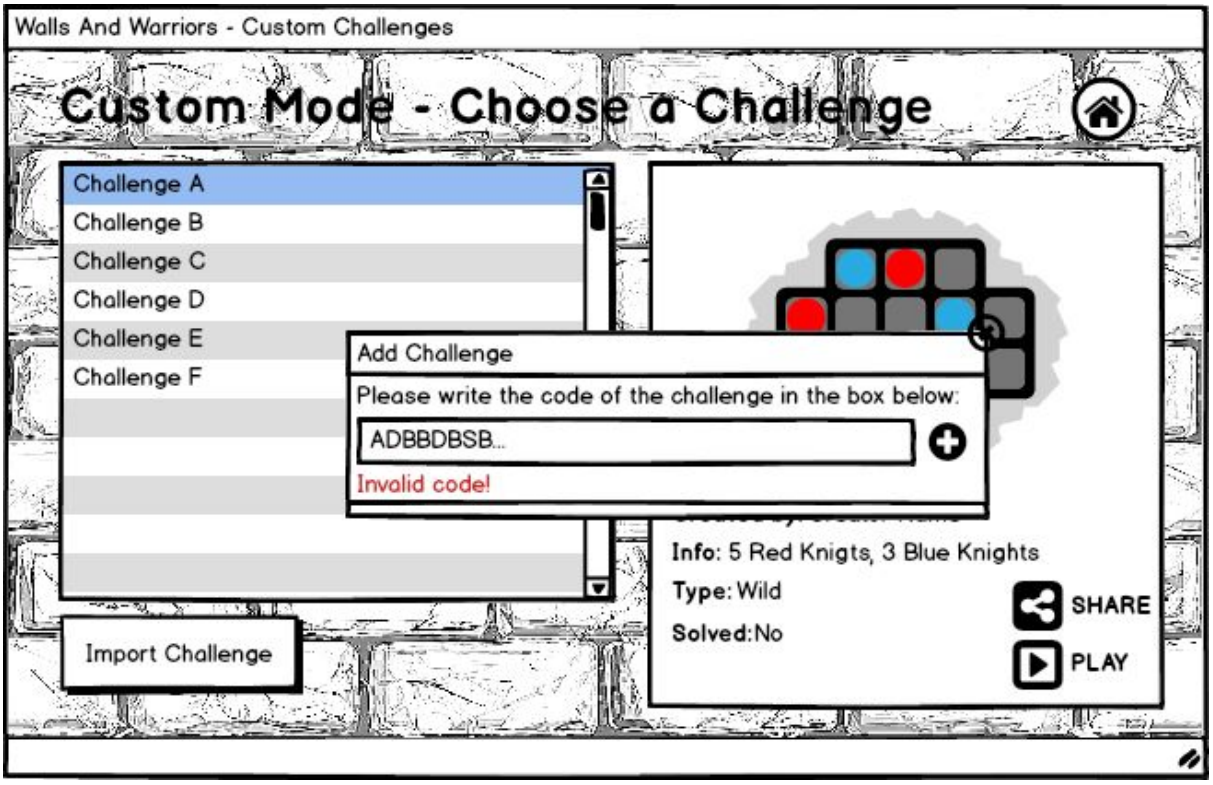


Figure 33: Invalid code warning

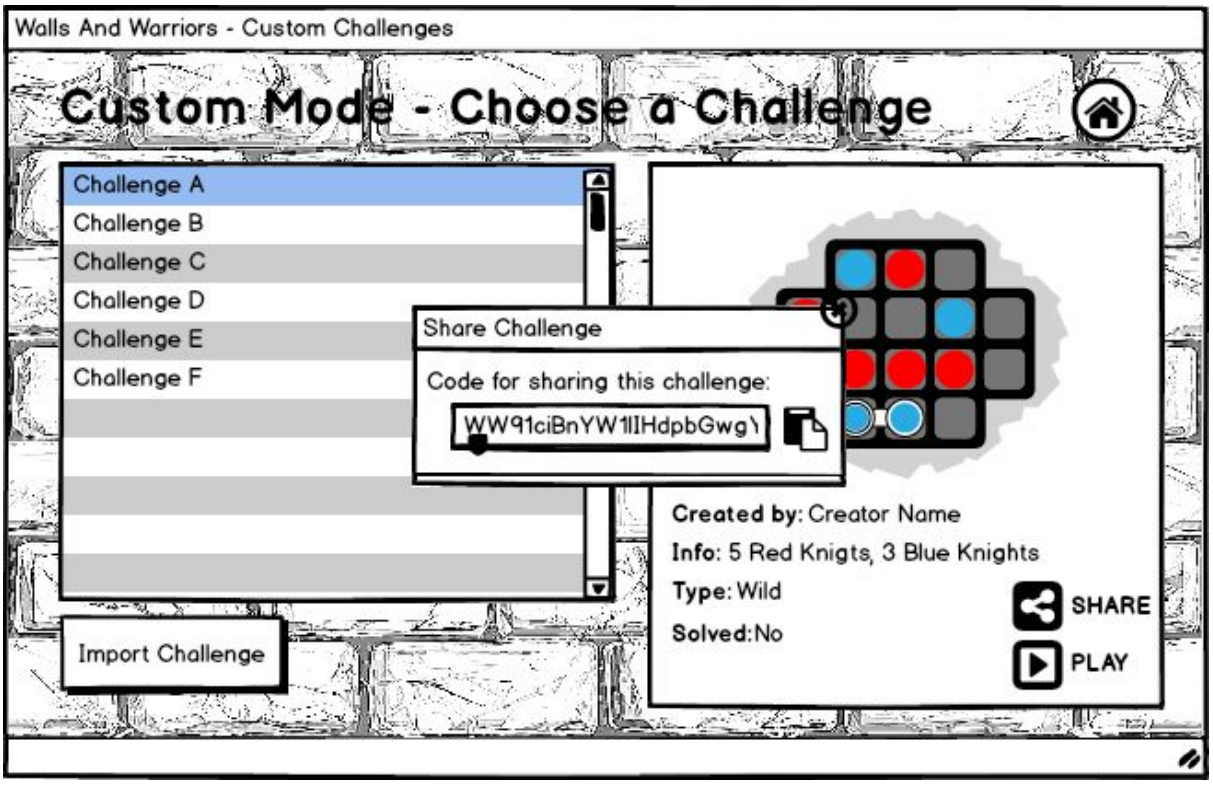


Figure 34: Share challenge

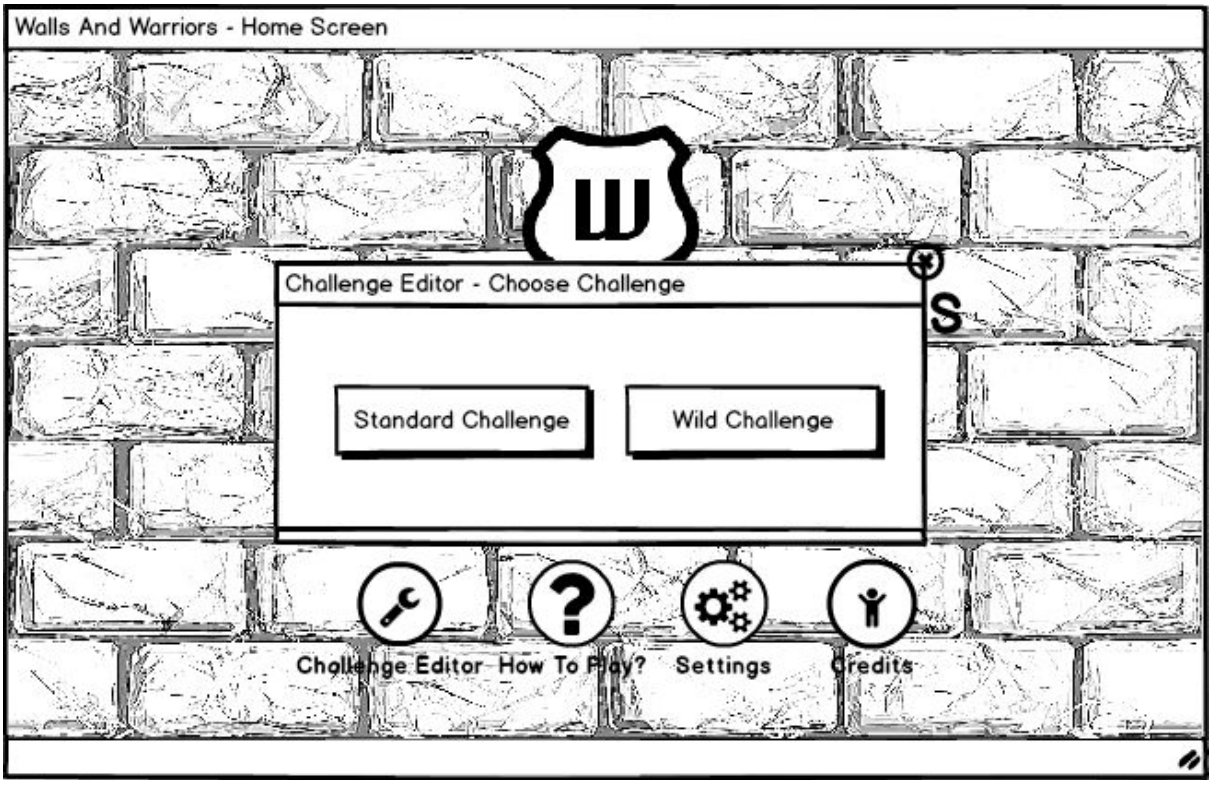


Figure 35: Mode selection for challenge editor

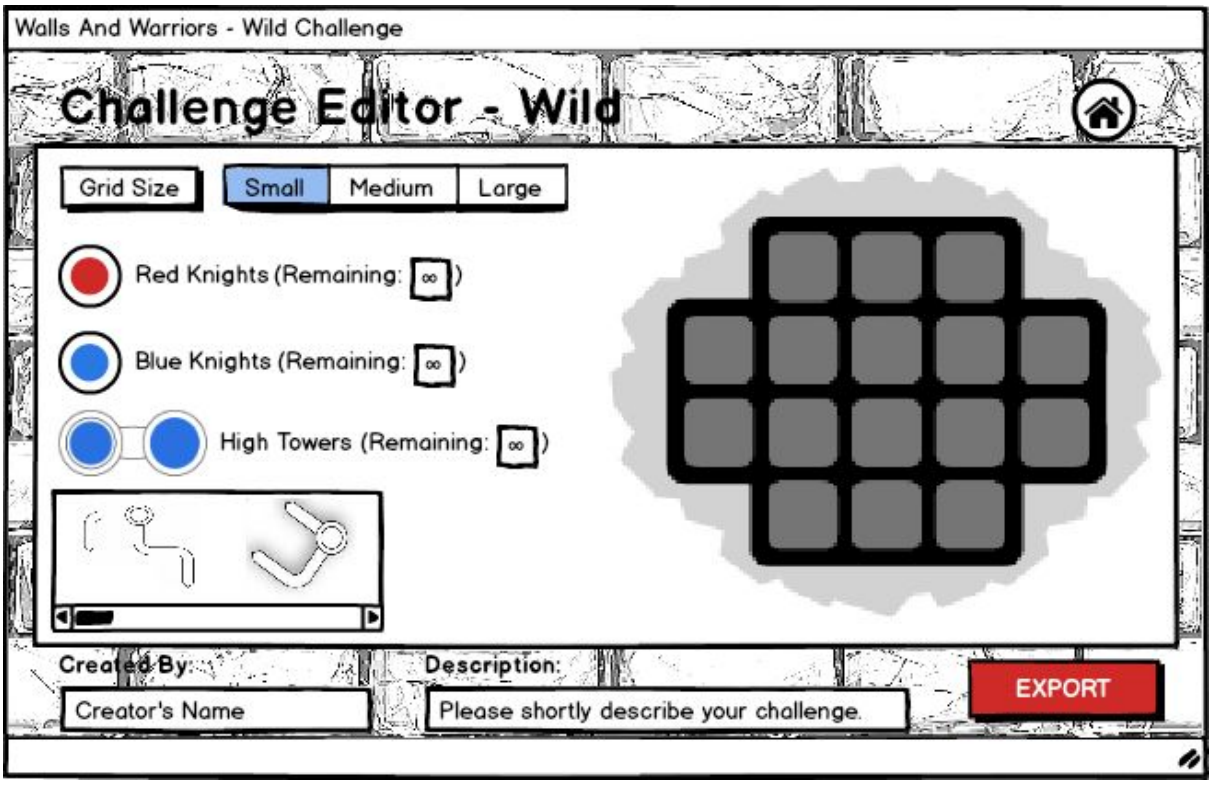


Figure 36: Wild challenge editor

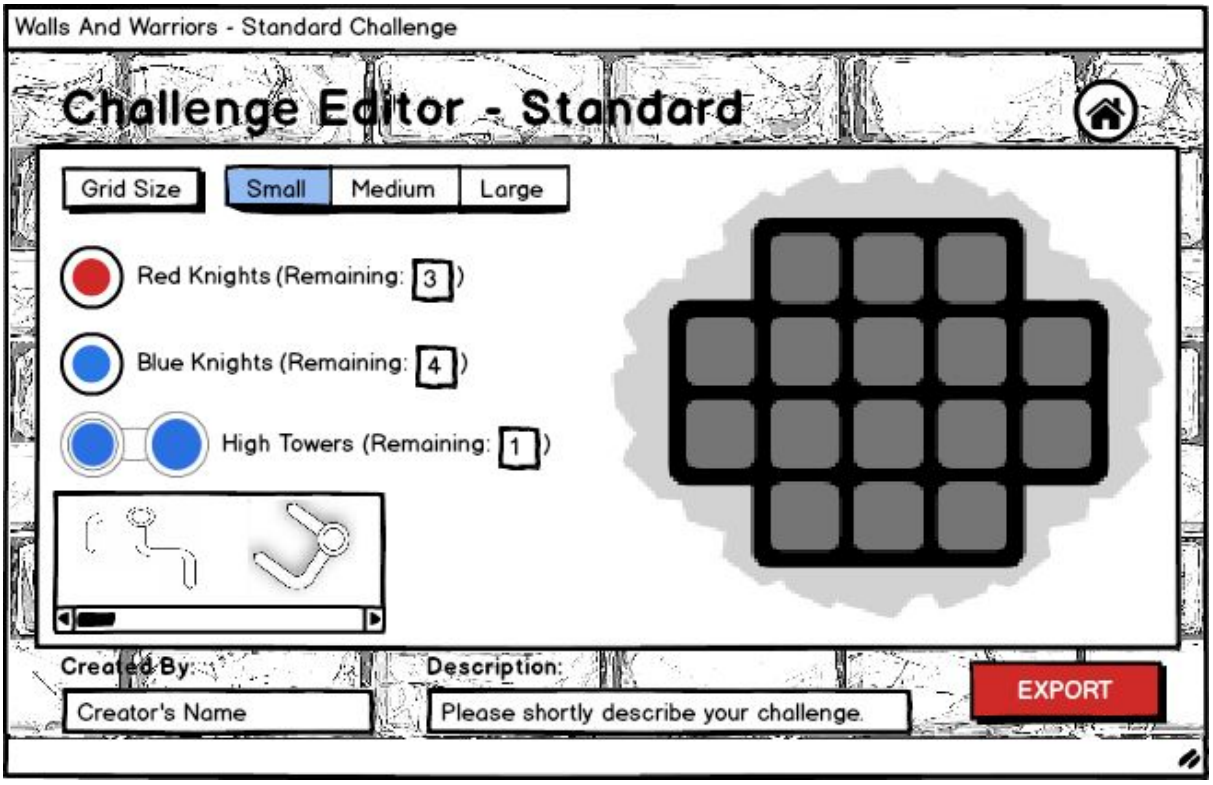


Figure 37: Standard challenge editor

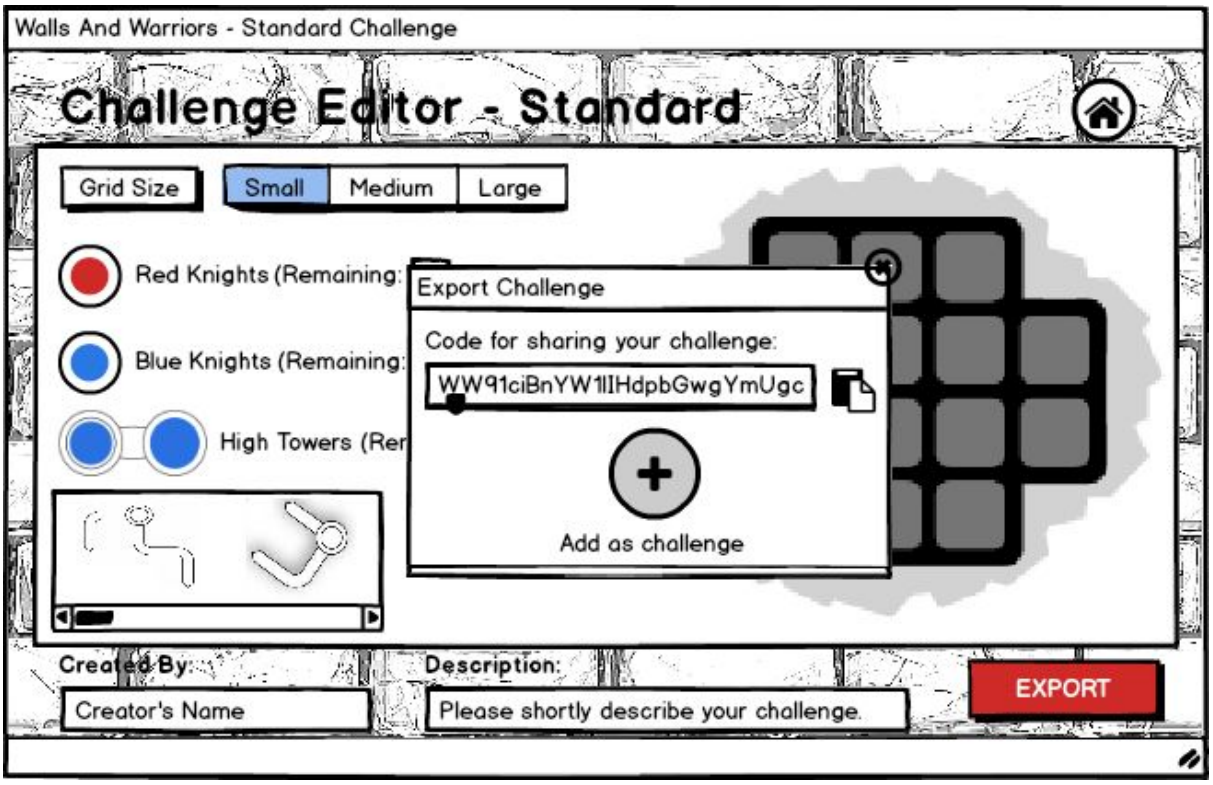


Figure 38: Export created challenge

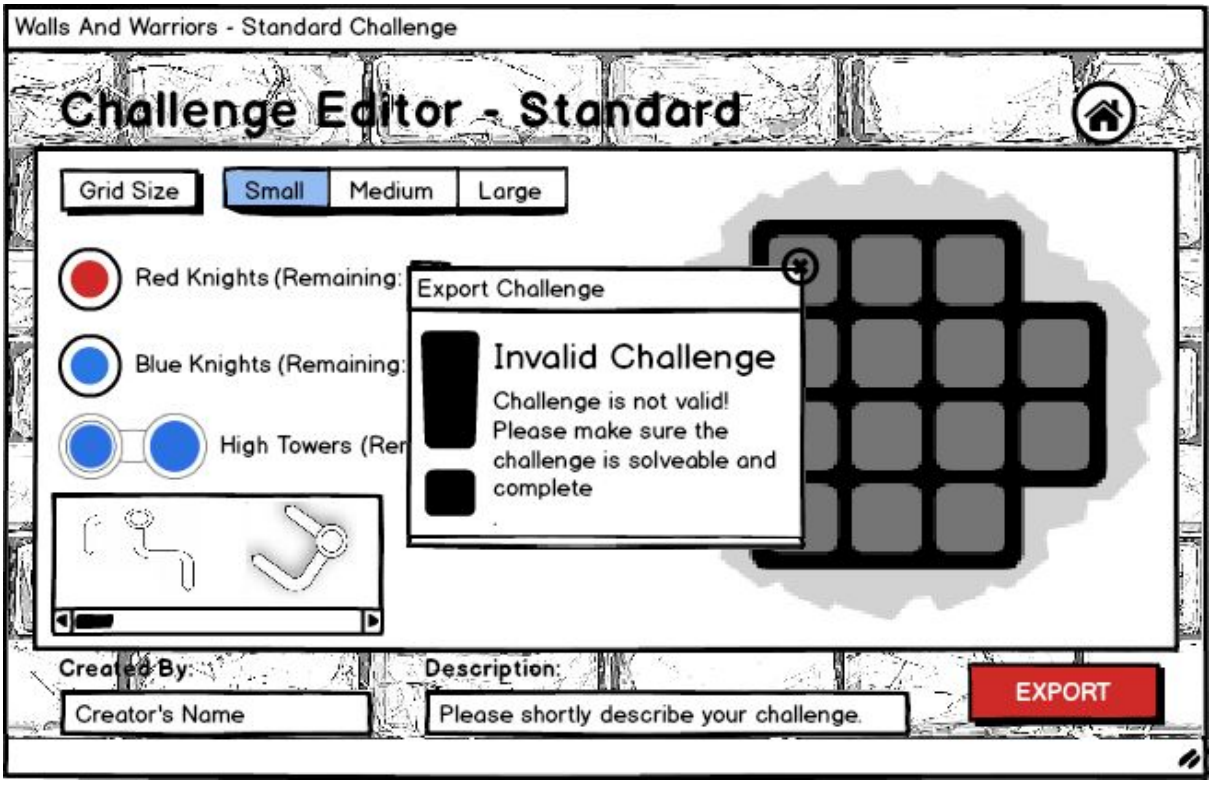


Figure 39: Invalid challenge warning

7. Improvement Summary

In our second iteration of the analysis phase, we have revised some of our diagrams and added/extended some of our requirements for the project. Here is a quick summary:

- To make the game playable by children and inexperienced players, we have introduced partially solved challenges that has some walls placed on the grid beforehand.
- We have added measurability (metrics) to all of our non-functional requirements.
- We have improved our definition and metrics about the non-functional requirement of extensibility.
- We have shortened our introduction section of our report for concision.
- We have revised our use-case diagram to reflect in-game interactions better.
- We have revised our activity diagram to reflect a simpler and more understandable flow.
- We have added a new sequence diagram for the process of checking the solution of a challenge.
- We have revised our state diagram to reflect the possible states of a wall piece in a better way.
- We have added a new state diagram for representing the current display status of the challenge screen.
- We have removed our sequence diagrams that addresses very basic interactions. (Learn how to play, see credits etc.)
- We have added cardinality constraints to our class diagram.

7. References

[1] Peeters, R. (2016, January). “The Story Behind the Creation of Walls & Warriors”. SmartGames.http://www.smartgamesandpuzzles.com. [Accessed: Oct 13, 2018]