Onitama

Final Report

Team Onitama

Alex Keyser

Caleb Lefcort

Colin McClelland

Course: CPSC 224 - Software Development

I. Introduction and Project Description

For our final project, we decided on the board game Onitama, a two-player strategy game that uses many ideas from chess to form a completely new and unique experience. In Onitama, each player has 5 pieces in total, 4 student pawns, and 1 master pawn per player on a 5x5 square board. What is unique about this game is how your pieces move. Your movement is decided by 5 random cards drawn at the start of the game. Each card has a different move set indicated on it which you can apply to any of your pieces. Each player always has 2 cards at a time, which rotate after you use one and switch with the other player. This adds a really unique strategy to the game where you know exactly what moves will be available to your opponent on your turn, and you can even prevent them from getting one move by holding it on your side.

For our implementation of this game, the main features we focused on were an accurate representation of Onitama, an intuitive interface, and a character/icon selection system. There were some limits that we encountered, as we could only make so many different sprites for the project at this time. We don't have a tutorial, but we provide the rules as an option to look at in case one doesn't know how to play.

This document is an in-depth look into our process of re-creating this amazing game, and the progress as well as the roadblocks that we encountered along the way. This serves as a look into our first real group project as computer scientists, as well as a documentation of what we've learned over this semester. We are incredibly proud of the project and are excited to keep fine-tuning it during the summer so we can have a final project that we are proud to share with others and hopefully put on our resumes.

II. Team Members - Bios and Project Roles

Alex Keyser is a 2nd-year computer science student who is extremely passionate about game development and new technological advancements in the world of computer science. He has little experience yet in the game development world but has been working on multiple projects in Unity, including a multiplayer tank game that he is working on with the ASM club. Alex's skills include C/C++, basic Python with API/database management, Java, x86 assembly, and C# with the Unity library. For this project, he worked on all of the art/sprites shown in the game that he based on the original game design. For the coding, he was responsible for the implementation of the cards, basic game logic that eventually evolved into what we have now, and slight UI work as well as lots of bug fixing.

Caleb Lefcort is a 2nd-year computer science student with an interest in software, web, and mobile development. He has experience with C++, Python, Java, x86 assembly, and Data Science. Caleb focused on model development i.e. game logic and classes maintaining the state of the game. Caleb also worked on the button functionality and interaction with the model. He hopes to continue development on the game during the summer.

Colin McClelland is a 2nd-year computer science student with broad interests in the field of computing including mobile app development, UI/UX design, and machine learning. He has experience in Java, C/C++, and Python for data science. In regard to this project he focused on designing and implementing a visually appealing and intuitive user interface. His additional contributions included testing the implementation of the game and ensuring its functionality, as well as planning the logic that drives gameplay. Colin hopes to be able to further explore the world of software development through a combination of personal projects and academic studies.

III. Project Requirements

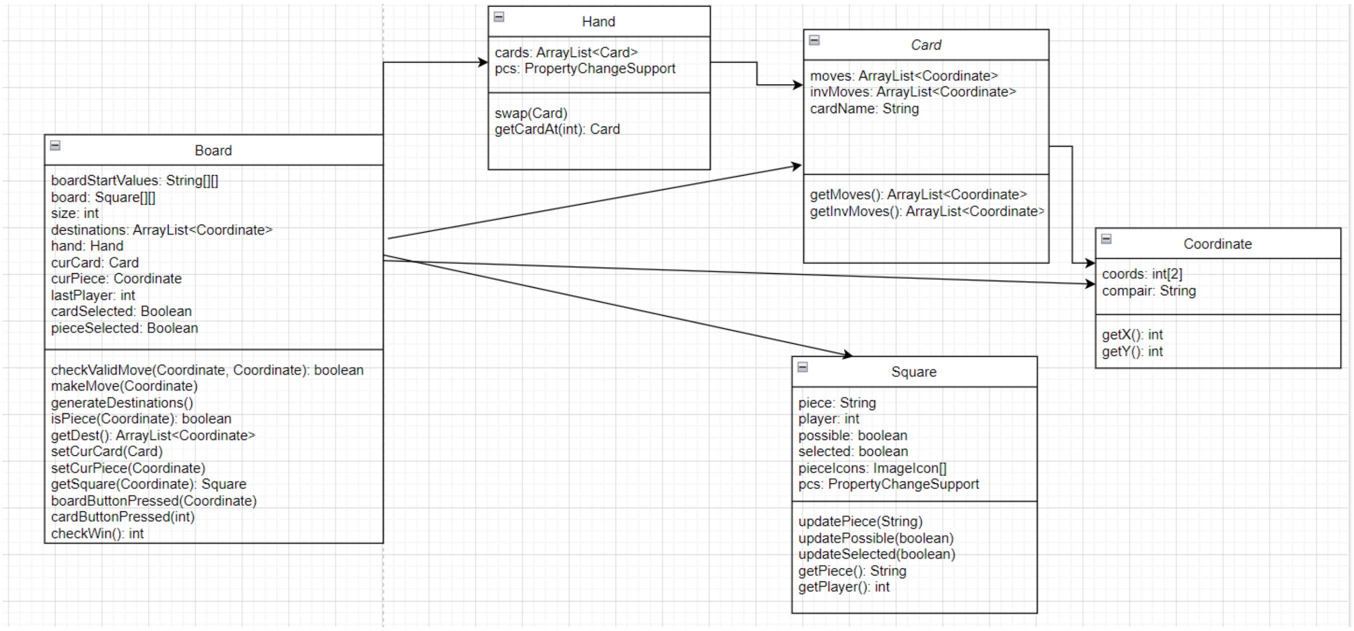
The requirements for this project were quite easy to define as we were trying to create as accurate a depiction of Onitama as possible in the provided time period. This meant taking a detailed look into the official rules of the game, and figuring out what could translate to our program and how. We settled on three main functional requirements to work towards, those being: cards and pieces are separate but work together, turn rotation and access to cards along with it, and the win conditions. Once these requirements were met, we had a functional game. Of course, that is not everything we wanted to do, as we ended up with more and more non-functional requirements as the project developed.

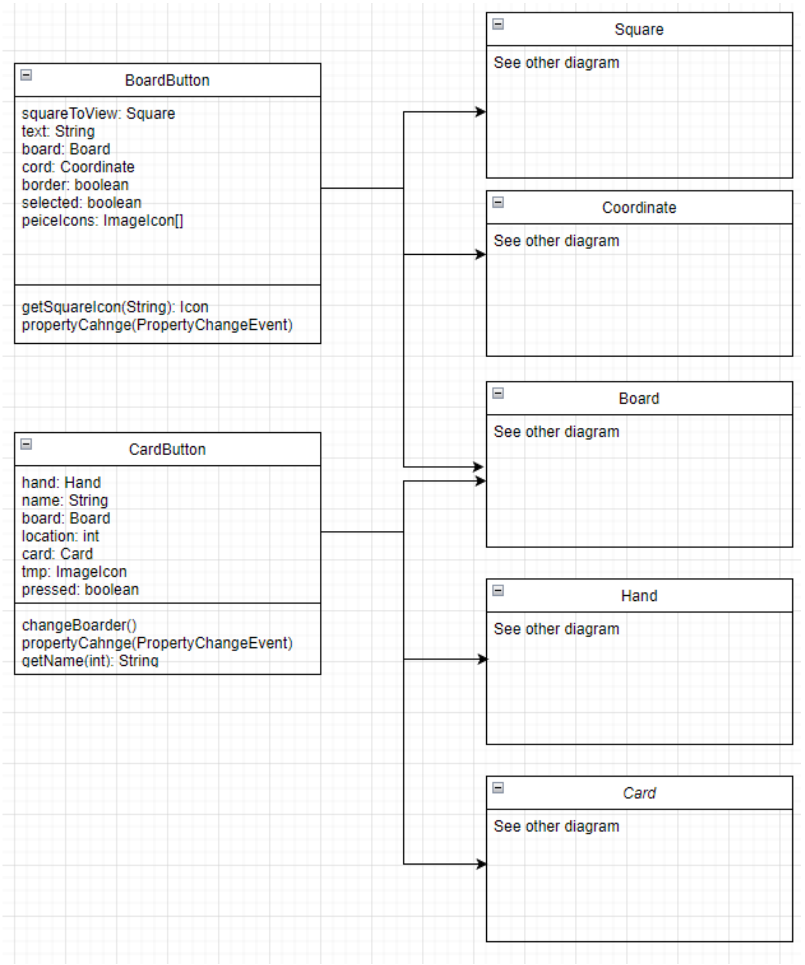
The non-functional requirements for this project were focused on making the game look and feel like you are playing the original game, as well as making it as intuitive as possible, to the point where someone who hasn’t played the game should be able to get the gist of it quickly. Our main requirements here were: custom art, character/icon selection, card movement around the board, moving Icons, and an easy-to-understand display. These all worked together to create a really beautiful and easy-to-get-into design that we were all really proud of.

IV. Solution Approach

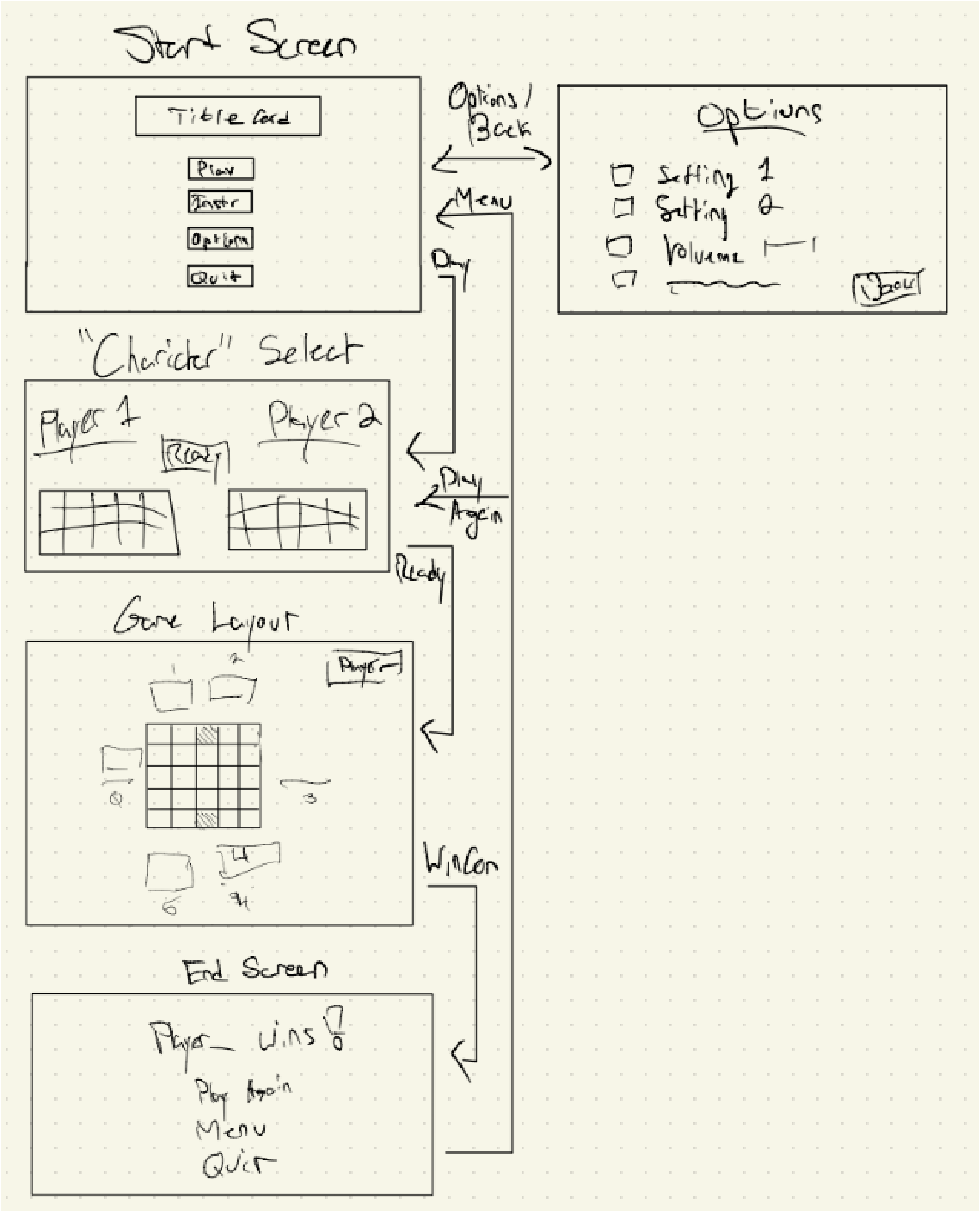
For the overall solution approach we made a clear separation between model, view, and controller. For the model we have the Board which is a 2d array of Squares. The majority of the game logic is run through this class and the state of the game is stored here. Also included in the model is the Hand class which is an ArrayList of Cards. The Hand holds the Cards that will be used in the game and changes their location based on the player's decisions. We then have the controller which has two parts. First we have a grid of BoardButtons with each button observing its respective Square for changes. The BoardButtons notify the Board when they have been clicked and the Board responds accordingly based on the location of the button and the state of the game. Once the Board has been updated the buttons change their appearance to match the Board. The CardButtons are the other aspect of the controller. The CardButtons notify the Hand and the Board of the Card selected by the player. The Hand and Board then update and notify the necessary listeners. The view and controller were very closely linked due to the nature of our project i.e. the buttons display the state of the game to the user. The GUI associated with the game is kept separate from the model code and passes information to the buttons such as the icon selected.

UML Class Diagram

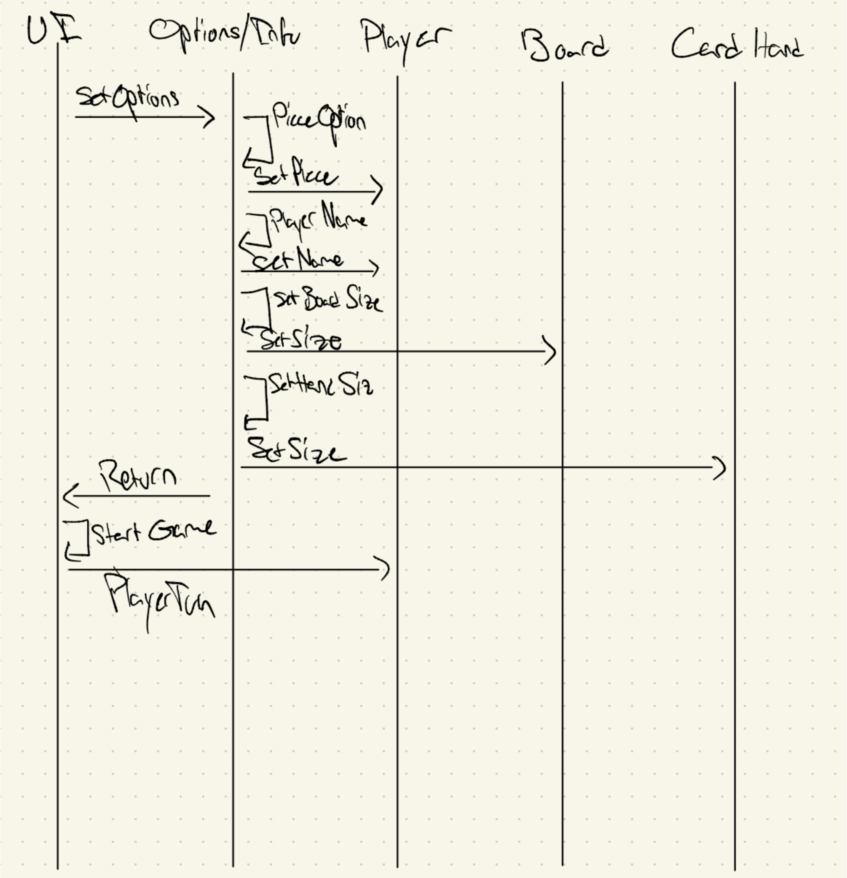


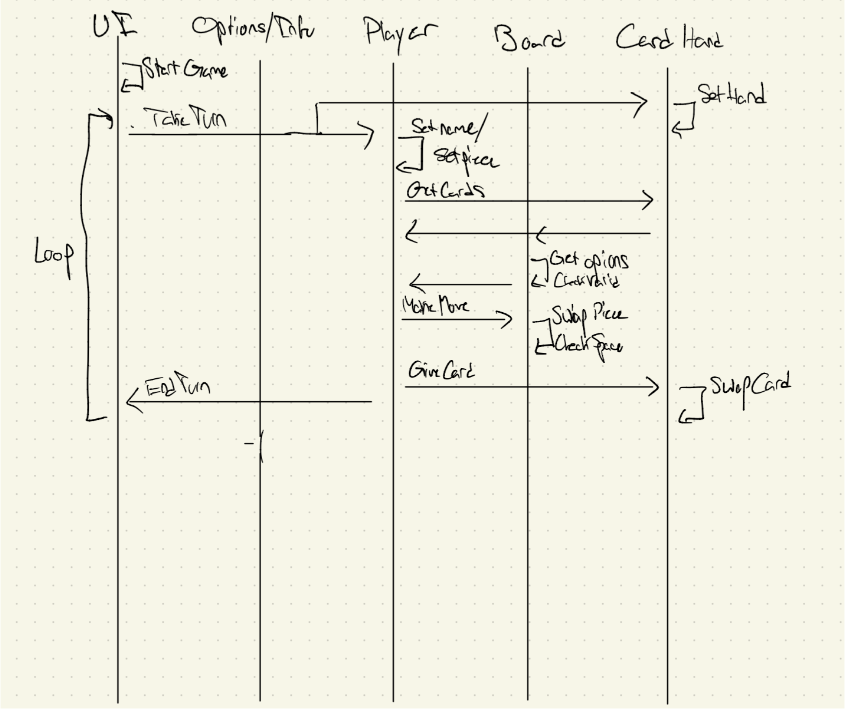


StoryBoard


Start Screen
Title Card
Play
Instr
Options
Quit
Options / Back
Options
Setting 1
Setting 2
Volume
Back
Menu
Play
"Charicter" Select
Player 1
Player 2
Ready
Play Again
Ready
Game Layout
Player
1
2
3
4
5
6
WinCon
End Screen
Player_ wins!
Play Again
Menu
Quit

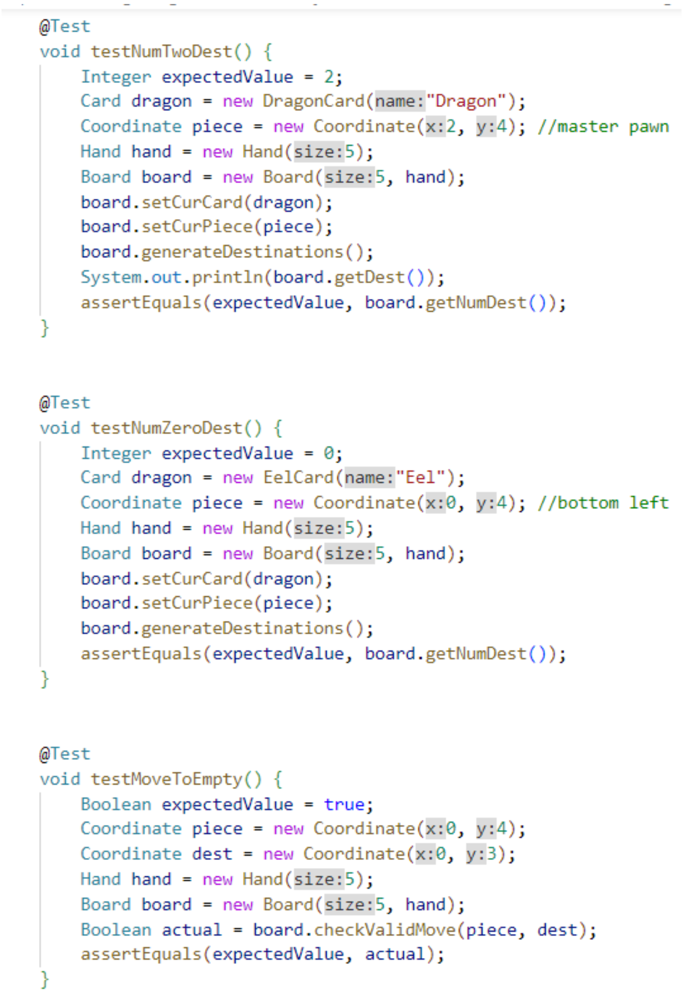
Sequence diagram





V. Test Plan

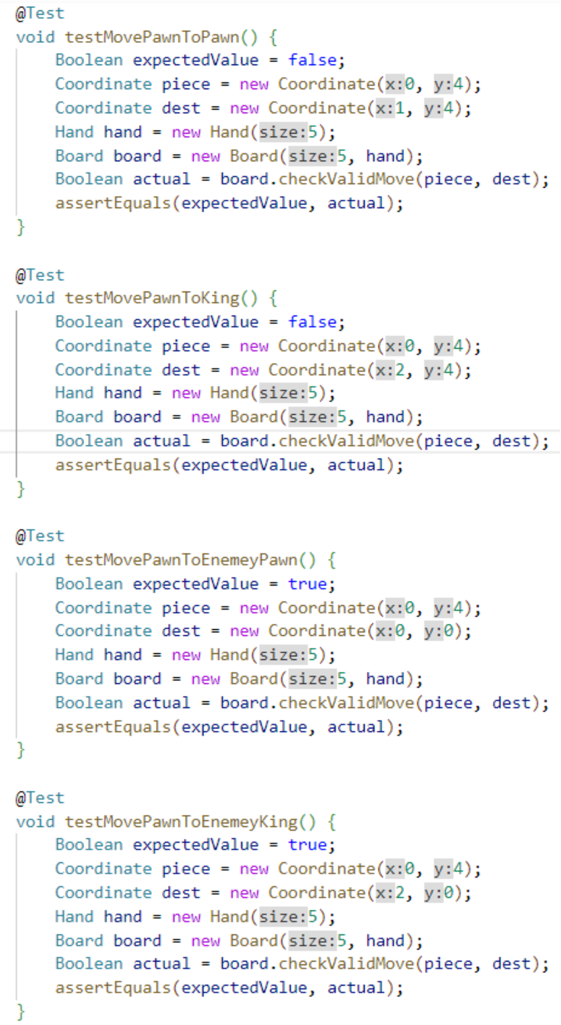
Testing our implementation of Onitama required several strategies. We mainly relied on informal testing which occurred throughout the entire development process. As changes to the existing code were made and new additions were integrated, we would run the game several times to ensure that it achieved the desired result and that any necessary changes or fixes were made accordingly. We routinely performed integration testing to ensure that the various components that comprised the game worked together. When it came time to merge the GUI frontend with the game implementation, we spent a significant amount of time making sure that when a player interacted with the various controllers, the model, and the view were updated as necessary. We also constructed unit tests to ensure that the game logic produced the expected results when presented with various conditions. As the game neared completion user testing also became central to our testing approach. This allowed us to generate feedback and make any adjustments as necessary. The feedback we received was mostly concerned with the interface because at this stage of development, we had already thoroughly tested the game logic. The input that we received allowed us to fine-tune the game such that it was more intuitive and user-friendly. Upon completion of the game, we revisited the requirements as provided by the project description and those that we had created ourselves. We carefully evaluated our build to make sure that we had met all the requirements and revisited any areas that we felt did not fully satisfy the condition. This process continued until we were certain that the game was ready to be completed.



```java
@Test
void testNumTwoDest() {
    Integer expectedValue = 2;
    Card dragon = new DragonCard(name:"Dragon");
    Coordinate piece = new Coordinate(x:2, y:4); //master pawn
    Hand hand = new Hand(size:5);
    Board board = new Board(size:5, hand);
    board.setCurCard(dragon);
    board.setCurPiece(piece);
    board.generateDestinations();
    System.out.println(board.getDest());
    assertEquals(expectedValue, board.getNumDest());
}


@Test
void testNumZeroDest() {
    Integer expectedValue = 0;
    Card dragon = new EelCard(name:"Eel");
    Coordinate piece = new Coordinate(x:0, y:4); //bottom left
    Hand hand = new Hand(size:5);
    Board board = new Board(size:5, hand);
    board.setCurCard(dragon);
    board.setCurPiece(piece);
    board.generateDestinations();
    assertEquals(expectedValue, board.getNumDest());
}


@Test
void testMoveToEmpty() {
    Boolean expectedValue = true;
    Coordinate piece = new Coordinate(x:0, y:4);
    Coordinate dest = new Coordinate(x:0, y:3);
    Hand hand = new Hand(size:5);
    Board board = new Board(size:5, hand);
    Boolean actual = board.checkValidMove(piece, dest);
    assertEquals(expectedValue, actual);
}
```



```java
@Test
void testMovePawnToPawn() {
    Boolean expectedValue = false;
    Coordinate piece = new Coordinate(x:0, y:4);
    Coordinate dest = new Coordinate(x:1, y:4);
    Hand hand = new Hand(size:5);
    Board board = new Board(size:5, hand);
    Boolean actual = board.checkValidMove(piece, dest);
    assertEquals(expectedValue, actual);
}

@Test
void testMovePawnToKing() {
    Boolean expectedValue = false;
    Coordinate piece = new Coordinate(x:0, y:4);
    Coordinate dest = new Coordinate(x:2, y:4);
    Hand hand = new Hand(size:5);
    Board board = new Board(size:5, hand);
    Boolean actual = board.checkValidMove(piece, dest);
    assertEquals(expectedValue, actual);
}

@Test
void testMovePawnToEnemeyPawn() {
    Boolean expectedValue = true;
    Coordinate piece = new Coordinate(x:0, y:4);
    Coordinate dest = new Coordinate(x:0, y:0);
    Hand hand = new Hand(size:5);
    Board board = new Board(size:5, hand);
    Boolean actual = board.checkValidMove(piece, dest);
    assertEquals(expectedValue, actual);
}

@Test
void testMovePawnToEnemeyKing() {
    Boolean expectedValue = true;
    Coordinate piece = new Coordinate(x:0, y:4);
    Coordinate dest = new Coordinate(x:2, y:0);
    Hand hand = new Hand(size:5);
    Board board = new Board(size:5, hand);
    Boolean actual = board.checkValidMove(piece, dest);
    assertEquals(expectedValue, actual);
}
```

VI. Project Implementation Description

Our team successfully achieved all of the functional requirements we initially intended to meet.

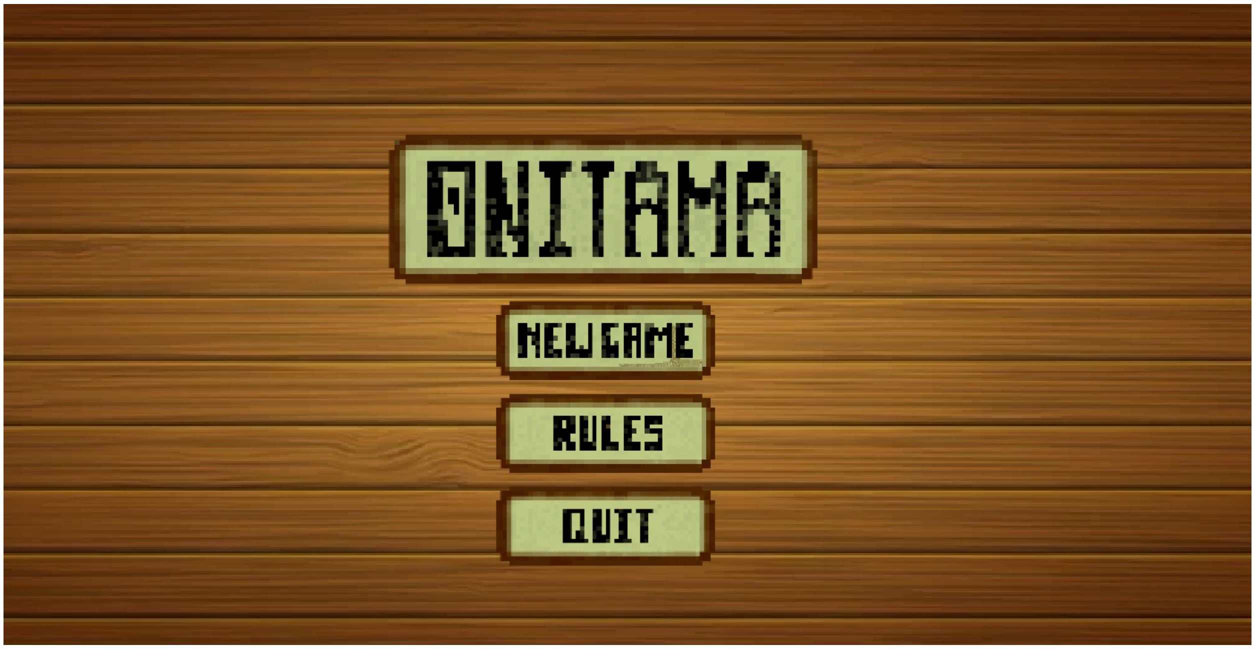
- Cards and pieces are separate but work together.
- Turn rotation and access to cards are fully functional.
- The win conditions are properly implemented and trigger the appropriate response.

We also had time to implement most if not all of our planned non functional requirements.

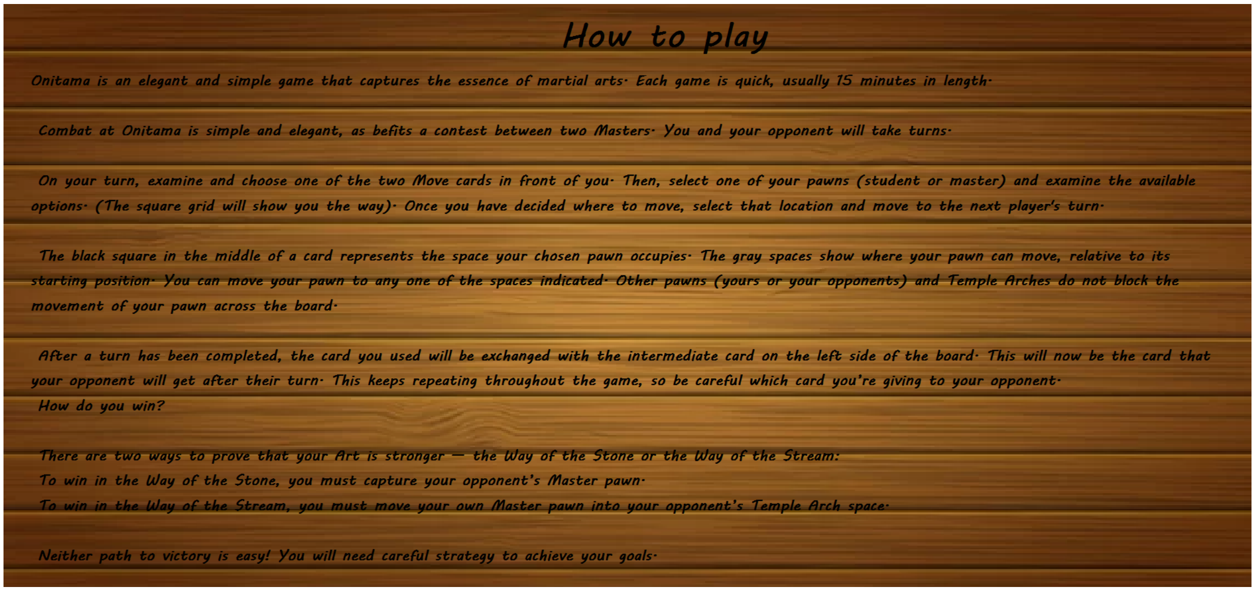
- Custom art for the pieces and cards.
- Character/icon selection for each of the players.
- Card movement around the board.
- An easy-to-understand display.
- A functional Splash Screen.
- A rules page.
- A Win Screen.

All of the elements of the solution approach were completed In a fairly efficient manner. The main modification that would improve our code would be simplification in certain areas. This would help if/when we decide to update the code in the future.

Splash Screen



Rules Page



How to play

Onitama is an elegant and simple game that captures the essence of martial arts. Each game is quick, usually 15 minutes in length.

Combat at Onitama is simple and elegant, as befits a contest between two Masters. You and your opponent will take turns.

On your turn, examine and choose one of the two Move cards in front of you. Then, select one of your pawns (student or master) and examine the available options. (The square grid will show you the way). Once you have decided where to move, select that location and move to the next player's turn.

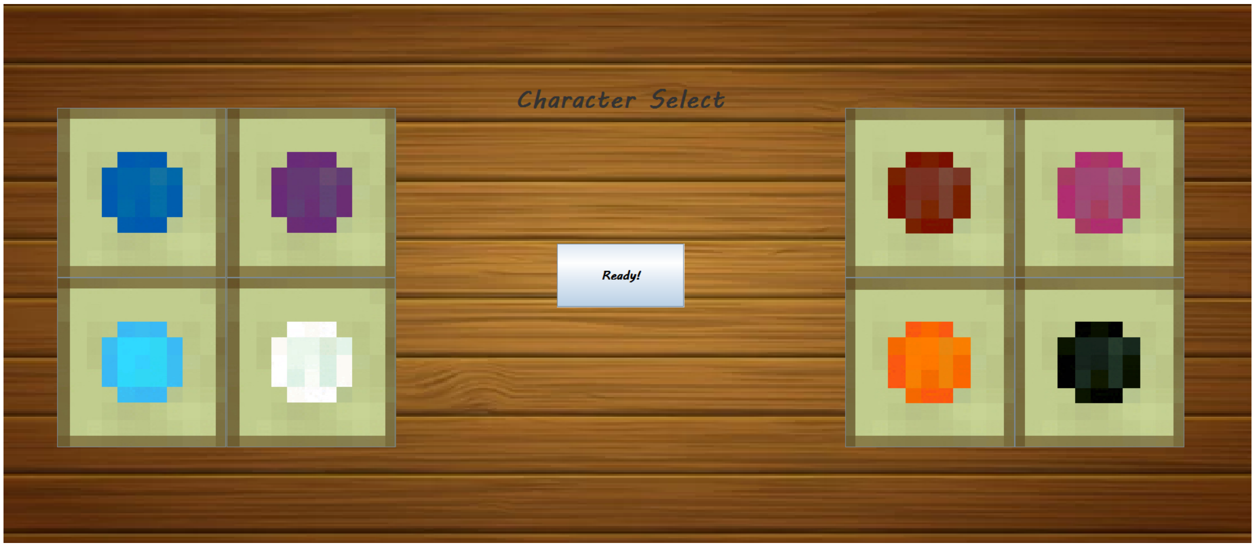
The black square in the middle of a card represents the space your chosen pawn occupies. The gray spaces show where your pawn can move, relative to its starting position. You can move your pawn to any one of the spaces indicated. Other pawns (yours or your opponents) and Temple Arches do not block the movement of your pawn across the board.

After a turn has been completed, the card you used will be exchanged with the intermediate card on the left side of the board. This will now be the card that your opponent will get after their turn. This keeps repeating throughout the game, so be careful which card you're giving to your opponent.
How do you win?

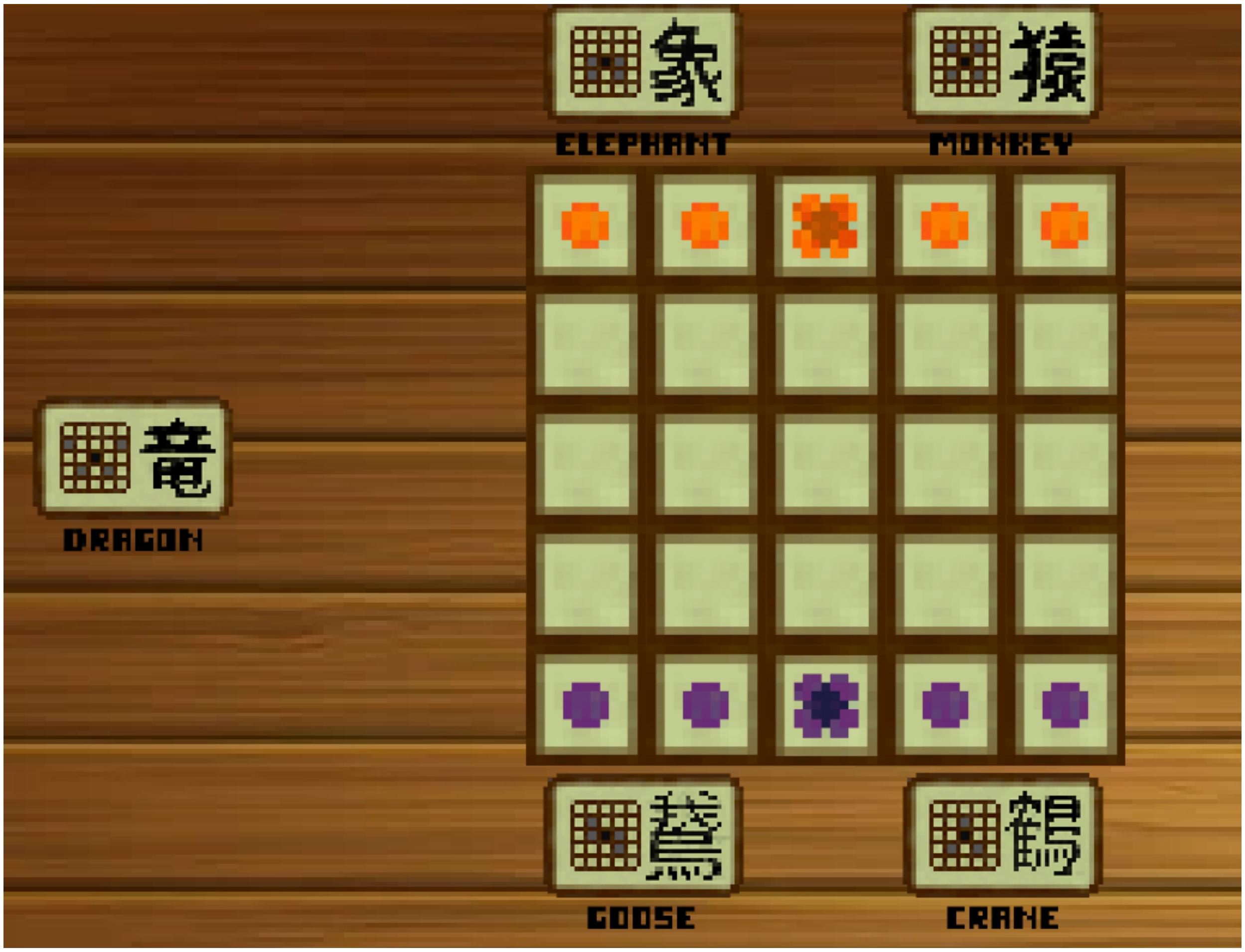
There are two ways to prove that your Art is stronger — the Way of the Stone or the Way of the Stream:
To win in the Way of the Stone, you must capture your opponent's Master pawn.
To win in the Way of the Stream, you must move your own Master pawn into your opponent's Temple Arch space.

Neither path to victory is easy! You will need careful strategy to achieve your goals.

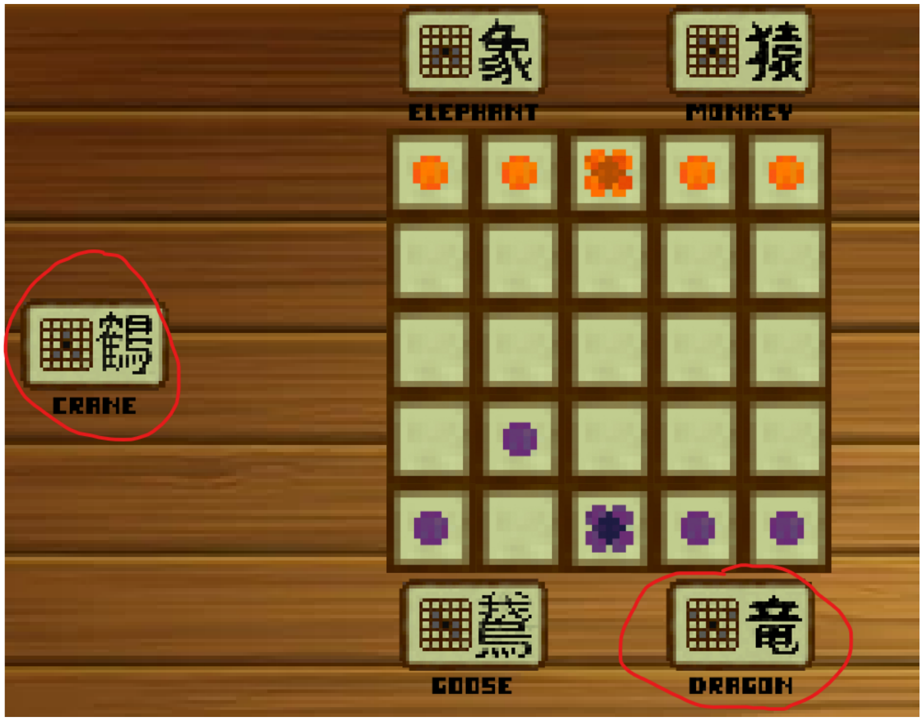
Character Selection



Easy-to-understand display



Card movement around board (compare to previous image)



Win Screen

To get a full idea of the functionality of the game try it here
https://github.com/GU-2023-Spring-CPSC224/final-software-dev-project-team-onitama

VII. Future Work

As it stands, our implementation of Onitama is currently a fully functional two player game. The game runs as intended and meets all of its requirements, however there are still additional improvements that could be made to advance the game even further.

GUI Enhancements: There are several GUI features that could be implemented to make the game more intuitive and customizable. A background change feature was partially implemented but required additional work and therefore did not reach the final version. Additionally, an indication as to which movement card the player selected would make the game more intuitive, especially for new and unfamiliar players.

Expansion Packs/Additional Movement Cards: Additional movement cards could be created similar to the expansion packs that exist for the original board game. These added cards would add variety and improve the replay value of the game.

Enhanced End Screen: The current implementation of the end screen is noticeably bare and feels somewhat lacking in terms of providing players with feedback and a sense of accomplishment. To improve upon this, we could include game statistics such as total time played, number of moves made, and pieces captured. This will allow players to review their performance and compare it with previous matches.

Packaging as Downloadable Application: In order to make the game more easily accessible, we could package the game as a downloadable application. This would allow more people to play our game and make it much more distributable.

VIII. Glossary

Board - A five by five or seven by seven grid similar to a chess board.

Cards - A list of possible moves.

Piece - The pieces of the game all have the same movesets that are determined by the cards. The only exception is the king which can lose the game if captured.

GUI - Graphical user interface.

IX. References

Emery, Brad. “Onitama Cards - Follow the Link to Find a Printable Version: Bordspellen, Spelletjes.” Pinterest, October 22, 2018. https://www.pinterest.com/pin/229894755963886905/.

Niebling, William. “Onitama Rulebook.” Scribd. Scribd. Accessed May 7, 2023. https://www.scribd.com/document/584656096/Onitama-Rulebook.

Adamson, Jack. “Onitama App.” Onitama App, July 16, 2021. https://onitama.app/#/.

mischa_u. “Onitama: Chess-like Abstract Strategy Boardgame.” Onitama: chess-like abstract strategy boardgame, December 22, 2020. https://www.lexaloffle.com/bbs/?tid=40893.