**CS 471 Team Project Final**

**Nine Mens Morris Game**

**Team DMQ**

**Brian Dunn, Brendon McCoy, Adam Quarnstrom**

CS471/571 Team Project

* **Section I. Team Organization and Buddy Rating**

1. Team Organization

Brendon is in charge of the GUI, Adam is in charge of the Logic, and Brian is in charge of Testing and AI. We split the project up this way during our first meeting. Brendon was most comfortable working on the GUI, whereas Adam really wanted to get into figuring out the logic needed for the game, and Brian felt that he wanted to become more skilled in testing and AI development so he wanted to focus on that. All that being said, the majority of this project was programmed during pair programming sessions that we had every week, so we all got a chance to work on different parts of the project together.

1. Buddy Ratings

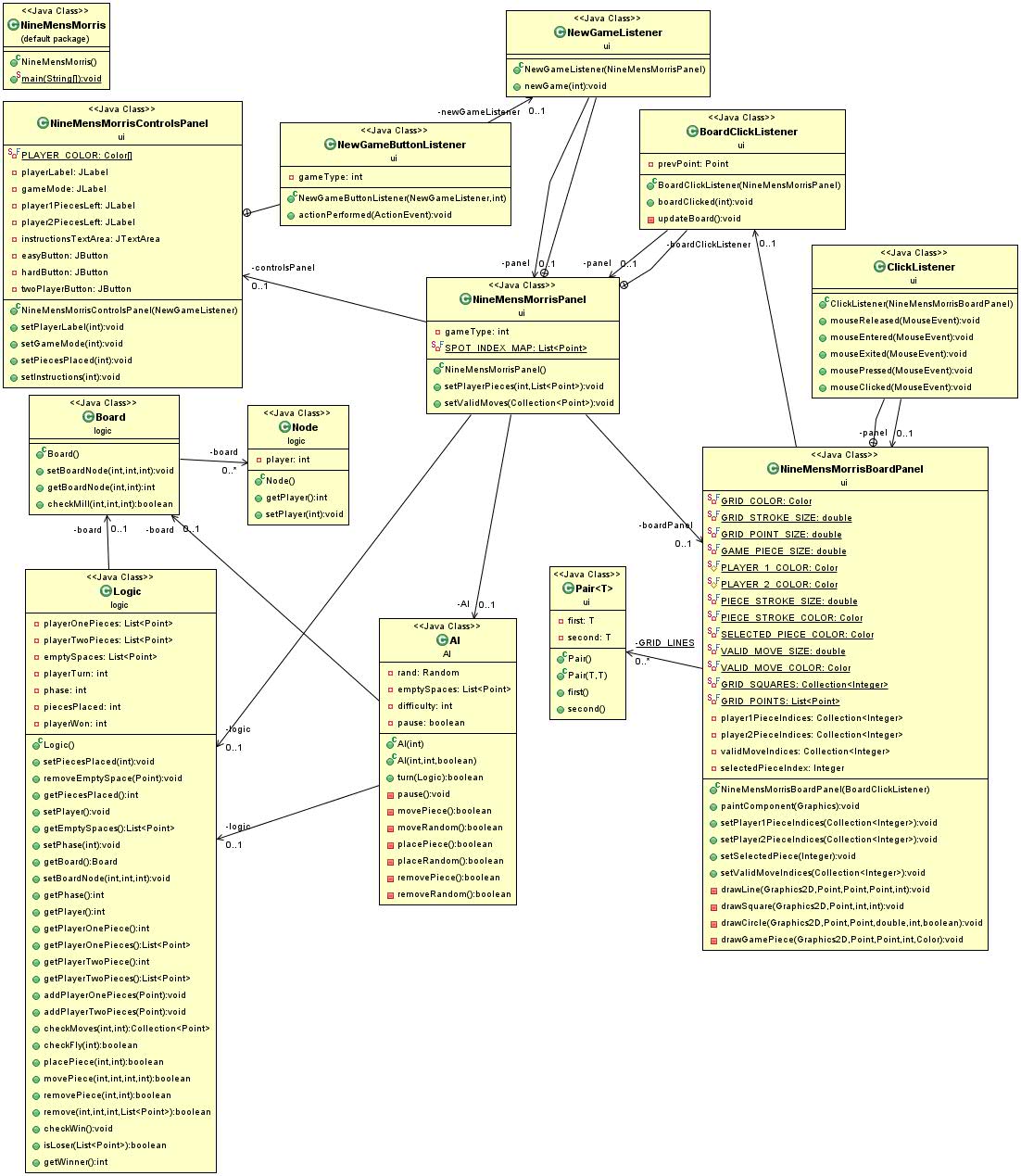
|  |  |  |  |
| --- | --- | --- | --- |
| **Team Members** | **Brian Rating** | **Adam Rating** | **Brendon Rating** |
| Brian Dunn | N/A | 1 | 1 |
| Adam Quarnstrom | 1 | N/A | 1 |
| Brendon McCoy | 1 | 1 | N/A |

1. Provide a list of your team meetings, including time, place, and meeting minutes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Meeting #** | **Time/place** | **Participants** | **Topics and decisions** |
| 1 | 9/15, 6:00 – 6:15, Engineering Bldg | Brian, Brendon, Adam | User Stories, Basic Design, Deciding meeting place |
| 2 | 9/24,6:00-6:15,  Metageek Lab | Brian, Brendon, Adam | Basic Board/GUI development. Board class and functions created. |
| 3 | 10/1,6:00-6:15,  Metageek Lab | Brian, Adam | Logic/GUI development.  Logic constructor, place pieces, beginning of move pieces function. GUI displays board, uses current functions. |
| 4 | 10/8,6:00-6:15,  Metageek Lab | Brian, Brendon, Adam | Finished move pieces, began check moves, and remove pieces, and check for win. GUI can use added functions. |
| 5 | 10/15,6:00-6:15,  Metageek Lab | Brian, Brendon, Adam | Finished check moves, remove, check win. GUI finished. |
| 6 | 10/22,6:00-6:15,  Metageek Lab | Brian, Brendon, Adam | Bugfixes, GUI beautification. |
| 7 | 10/29,6:00-6:15,  Metageek Lab | Brian, Brendon, Adam | Discuss AI implementation. Decided on a priority based system. |
| 8 | 11/5,6:00-6:15,  Metageek Lab | Brian, Brendon, Adam | Discuss extras for GUI, like displaying remaining pieces during placement phase. Also decided to have an easy and a hard version of the AI |
| 9 | 11/12,6:00-6:15,  Metageek Lab | Brian, Brendon, Adam | Discuss final report and state of testing. |
| 10 | 12/3,6:00-6:15,  Metageek Lab | Brian, Brendon, Adam | Finish going over testing, finalizing report. |

* **Section II. Design**

1. Summarize the user-interface design of your program, using a combination of screenshots and textual descriptions.
2. Provide a class diagram that captures the main classes and their relationships in your final program



1. Describe the algorithm for determining when a mill is formed based on the data structures used in your program.

The checkMill algorithm takes in the ordered pair (Square, Point) of which space you're checking in the 2D array used to hold the board spaces, and the player who created the mill. The algorithm simply checks that if the spaces on either side(using modulus to wrap around) to see if the three nodes match the given player, and if so it returns true, otherwise it returns false. Since the odd numbers correspond to the middle spaces on the board, only when an odd point is given, will it check the column it is in for a mill, otherwise it checks the row it is in. -Adam, Brendon

1. Describe the algorithm for determining when the game is over and who is the winner based on the data structures used in your program.

The checkWin algorithm passes the list of player pieces of whoever is the current player to a small function that was refactored to it's own function as it was duplicated in the original checkWin method. The isLoser function checks two conditions: if the current player has less than three pieces, or if the current player has no possible moves left. The latter is checked by iterating through the player's pieces and calling the check moves function and checking if, for each piece, the returned list of possible moves is empty. If it is for all pieces, then the function returns true, setting the game phase and playerWon variables to 4 and the winning player respectively. -Adam

1. Describe the detailed design of the computer opponent in your program.

We designed the computer opponent (AI) using a priority system, broken down by 3 phases, placing pieces, moving pieces, and removing pieces. For placing pieces, the first priority is to place a piece where it will create a mill for the AI. The next priority if a mill can't be created is to block a player from creating a mill, if possible. The next priority is to then attempt to place a piece next to one of the AI's existing pieces in the hope of being able to create a mill on the next turn. If none of those placements are possible, then just place piece in a random spot.

For moving pieces, the first priority is to move a piece into position to create a mill. If not possible, then next priority is to move to a position that will block the player from forming a mill. If not possible, then attempt to move a piece that is already within a mill so that it may be possible to create that mill again on the next turn. Finally, if none of those moves were possible, then just move randomly.

Finally, for removing pieces, the first priority is to remove a player's piece that is blocking the AI from creating a mill. If that is not the case, then the next priority is to remove a player piece that, if moved next turn, would possibly create a mill for the player. The final priority is to remove a random player piece. -Brian

The AI tests for placing pieces is similar to the test criteria for moving pieces. The tests are done with a seeded AI since the first placement of a piece is random no matter what, as all the other criteria the AI looks at aren't available, so it defaults to random. The tests cover the AI placing a piece first at random, then it prioritizes the following in descending order of priority: Creating mills, Blocking opponent's mills, Placing Adjacent, and finally at Random. A couple of test cases to show that it does do these. We have it place a single piece, test that it places a mill if it can, if it blocks an opponent's mill if it can, adjacent to it's own pieces if it can. The priority is tested by having both a mill to be blocked, and a mill that can be formed, and the AI does make the mill over blocking an opponent. -Adam

* **Section III. System Testing**

1. Describe the test requirements for the implementation of the computer opponent.

The testing of the computer opponent was difficult due to some random behavior that it has. We started testing by playing the game with each part of the AI implemented. For example, the first implementation was where the AI places pieces on the board in the first phase at random. We then played the game and verified that the AI succeeded in placing pieces in available spaces on the game board. We then tested random move and random removing of pieces by the AI, again by playing the game. It wasn't until later that we designed automated testing for the AI methods.

We needed to create tests for each of the priority cases laid out in our design of the computer opponent. So there needs to be tests where during the placing pieces phase where the AI will create a mill, block a player from forming a mill, place a piece next to one of their own, and places a piece at random. Also, the priority needs to be tested, where if it is possible for the AI to place a piece to create a mill or could place a piece to block a player mill, it should create a mill.

For testing of moving pieces, there should be a test for AI moving a piece to form a mill, block player from creating mill, moving a piece from within its mill, or randomly. Again the priority needs to be tested, to where if it is possible to block the player from creating a mill, it gets done instead of moving randomly.

Finally, we need tests for the AI's removing of pieces. Like the other tests, there should be individual tests for AI removing a player piece that is blocking it from forming a mill, close to forming a player mill, and removing random player pieces. Also, like the other tests, the priority needs to be tested as well. -Brian

1. Describe the detailed steps of two system tests where a human player plays against the computer player. One test demonstrates a complete game where the human player is the winner; the other test demonstrates a complete game where the computer player is the winner.

Every member must be involved in at least one of the above testing tasks.

* **Section IV. Lessons Learned**

**Adam Quarnstrom**

1. What did you personally gain from the project?

I have gained a good understanding of AGILE development, and a good experience of working with a team on a project, which isn't something you get a lot of in the CS program here. I also gained a better understanding of testing. I was a little bit lacking in that department, but I am much better at it now.

1. What does your program do well, and what could your program do better?

I think our program is pretty solid. I know the UI could probably be better but I think it's good as is. Otherwise, the only thing I can think of that could be way better is either the AI or the finding moves function. The finding move function is a bit of a mess, a lot of if/then statements, but we couldn't think of a better way to do it, and at least it works. The AI is pretty simple as far as AI goes. It works, and can win, especially if you're not playing well, but I know we could make a better AI if we had to, as this one only looks at the board as is to make decisions, it doesn't try to think ahead.

1. How could you improve your development process if you develop a similar game from scratch?

I think the development process could be improved just by the fact that we now have some AGILE experience. It was difficult to get into at first, because it's not at all how you do projects up until this point. Now that we all have done it, it would be easier to do again, easier to get started and not be fairly slow, or not falling back on the development process we are normally used to using.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_