Team Ninepenny Marl

Matthew Gillatt
Zakk Lambertsen
Matt Weaver

Table of Contents

Section I. Team Organization and Buddy Rating

Team Organization

Team Member	Tasks
Matthew Gillatt	Creation of graphics resources, Playtesting, Meeting documentation, Creation of Node class
Matt Weaver	AI, Test package, Creation of Board, Player, and AI class, Unit testing
Zakk Lambertsen	GUI, Creation of GUI and Game class, Catching input and drawing graphics

Buddy Ratings

Giver	Teammate 1	Teammate 2
Matthew Gillatt	Matt Weaver: 1.0	Zakk Lambertsen: 1.0
Matt Weaver	Matthew Gillatt: 1.0	Zakk Lambertsen: 1.0
Zakk Lambertsen	Matt Weaver: 1.0	Matthew Gillatt: 1.0

Section II. Summary of User Stories

Description of user story (author)	Acceptance criteria	Tasks for each user story
1. As president of the Western Idaho Ancient Boardgame Club, I would like to be able to play the game Nine Men's Morris against another club member. (Matt)	Game is Nine Men's Morris (24 spots on board, 9 pieces per player, 3 phases, etc). Game accurately enforces all game rules (allows valid moves and blocks invalid moves). Game has an easy to use GUI.	1. Build basic game architecture, to track board layout and state 2. Implement logic to enforce game rules 3. Build simple GUI for game
2. As a lowly soldier in Rome's 17th legion, I desire to demonstrate my tactical abilities to my commander, in order to secure a promotion. Since the command	User can choose to play against AI. AI appears to make an effort to win.	1. Create a simple AI for game

hierarchy seems fond of Nine Men's Morris, I want to master that game; however I have never played. I do not want to embarrass myself in front of my cohorts, so I want to be able to practice against some sort of magic, invisible player of a relatively low skill level. (Matt)		
3. As a player suffering for amnesia I sometimes forget where I clicked. I would like each the piece I am trying to move during phase 2 to be highlighted so I don't forget. (Zakk)	After clicking a piece to move it, the piece is marked. After the move, the mark goes away. Only pieces that can be moved get marked.	1. Implement a way of marking pieces when they have been selected to move
4. As a person with poor short term memory I can't keep track of how many pieces I have. I would like a counter to show how many pieces I have left. (Zakk)	During phase 1, the number of pieces left to place is clearly displayed. After phase 1, the number of pieces left on the board is clearly displayed.	1. Show a counter in the GUI for how many pieces are left to place for each player 2. Change counters so that when all 9 piece have been placed, it shows how many pieces remain on the board for each player.
5. As a learning player, I hate it when I click on spaces and it does nothing. If I do something wrong, it needs to tell me and explain what's wrong nicely, no computer error messages. (Matthew)	The game tells the player what they did wrong when an illegal move is made. The message is displayed in a status bar (as opposed to a dialog or something like that)	1. Display status bar, print illegal moves on status bar
6. As an old user, I don't like running programs over again to make them do something else. There should be buttons that make it easy to play again, or choose options. Games shouldn't have complicated menus. (Matthew)	There should be a new game button. When clicked, it must clear out any existing game state (pieces on board, etc) and start a new game. Options, like “2 player or AI” should be easy to find, not hidden in menus.	1. Add buttons for new game and exit. 2. Add radio buttons for options, such as 2-player or AI opponent.

Meeting #	Time/place	Participants	Topics and decisions
1	09/17 metageek lab	Matthew, Matt, Zakk	Organization, creation of basic classes
2	09/25 metageek lab	Matthew, Matt, Zakk	Schedule change, refactoring of Node, GUI building
3	09/30 metageek lab	Matthew, Matt, Zakk	GUI graphics implementing, checking mouse input
4	10/07 metageek lab	Matthew, Matt, Zakk	Schedule change, graphics changing
5	10/13 metageek lab	Matthew, Matt, Zakk	Completing features and playtesting
6	10/14 metageek lab	Matthew, Matt, Zakk	Organized bug list and feature requests
7	10/20 metageek lab	Matthew, Matt, Zakk	Finalize bug fixes, organized report and class presentation

Section III. Summary of Test-Driven Development and Refactoring

Test #	Description of test case (test input and oracle)	User story # and Task #	Developer(s)

1	n0.setRight(n1) -> n1.getLeft() == n0;
	n1.setLeft(n0) -> n0.getRight() == n1;
	n0.setDown(n1) -> n1.getUp() == n0;
	n1.setUp(n0) -> n0.getDown() == n1;
		User story 1, task 1	Matthew, Matt

2	(n1, n2, and n3 are in a horizonal line). If they all have the same owner/player, then n1.mill(), n2.mill(), and n3.mill() == true. Likewise, if they are in a vertical line.	User story 1, task 2	Matthew, Matt
3	(n1, n2, and n3 are in a horizonal	User story 1, task 2	Matthew, Matt

	line). If one of them has a different owner then either of the other 2, then mill() == false.		
4	n1's coordinates are (30, 30). (60, 30) -> n1.isInRegion() == true; (Inside region) (30, 60) -> n1.isInRegion() == true; (Inside region) (61, 30) -> n1.isInRegion() == true; (Outside region) (30, 61) -> n1.isInRegion() == true; (Outside region)	User story 1, task 3	Matt, Zakk
5	Game starts -> player's hasAvailableMoves() == true; Place <=9 pieces -> hasAvailableMoves() == true; Remove 6 pieces -> hasAvailableMoves() == true; Remove 1 more piece -> hasAvailableMoves() == false;	User story 1, task 2	Matt, Zakk
6	During phase 1, spot 0 is clicked by player1 -> Game.board.getPiece(0).getPlayer() == player1	User story 1, task 3	Zakk, Matthew
7	During phase 2, player1 clicks piece on spot 0, which they own -> Game.board.getPiece(0).isSelected() == true	User story 3, task 1	Zakk, Matthew
8	Player1 doing game.placePiece(0) -> node0.getPlayer() == player1	User story 1, task 1	Zakk, Matt
9	Player1 doing game.placePiece(0) when that spot is occupied -> exception	User story 1, task 1	Zakk, Matt
10	Player1 doing game.placePiece(0) out of turn -> exception	User story 1, task 2	Zakk, Matt
11	Player1 does game.placePiece(0) and creates a mill ->	User story 1, task 2	Zakk, Matt

	Game.expectedMove == Remove		
12	Player1 does game.placePiece(0) when the expected move isn't Place -> exception	User story 1, task 2	Zakk, Matt
13	Player1 does game.removePiece(0) on an opponent owned piece -> node0.getPlayer() == null	User story 1, task 1	Matt, Matthew
14	Player1 tries to remove their own piece or remove a piece from an unoccupied spot -> exception	User story 1, task 2	Zakk, Matthew
15	User selects AI opponent for game -> Game.player2 == AI.me	User story 6, task 1	Matt, Matthew, Zakk
16	User clicks "New Game" button -> old game != new game	User story 6, task 1	Zakk, Matt
17	User clicks "2 Player Button -> old game != new game, AIMode == false	User story 6,task 1	Zakk, Matt
18	User clicks "Computer" Button -> old game != new game, AIMode == true	User story 6, task 1	Zakk, Matt
19	User clicks on node 0 to place piece -> node 0.player == player 1	User story 1, task 3	Zakk, Matthew

Refactoring #	**Description of the refactoring (problem and solution)**	**Developer(s)**
1	Game.movePiece(Player, ...) smells funny, since it forces the GUI to track the current player, despite Game already tracking that. Refactored method/calls to not include Player argument.	Matt, Zakk
2	Game.players[] never really gets passed or used as a single unit; Player objects are always used individually. Replaced players[] with Game.player1 and Game.player2, since using an array just adds	Matt , Matthew

	unnecessary confusion.	
3	Using ints to represent colors in Player leads to someone confusing method signatures and code. Changed colors to an enum, to make code more readable.	Zakk, Matt
4	Tracking game phases is confusing and seems to be leading to spaghetti code. Switched to “expected move” model to simplify code.	Matt, Matthew
5	Having main program code in the same location as testing code is causing clutter. Moved all main program stuff into “main” package and all testing stuff into “testing” package, to help organize this.	Matthew, Zakk
6	AIPlayer doesn’t really make sense as a subclass of Player, in practice. Changed name to AI and switched from “is-a” to “has-a” relationship with Player.	Matthew, Matt, Zakk
7	Accessing private/protected variables with a <Class>.Test class smells funny and clutters up code with odd testing stuff, but having access to these variables is very convenient for testing. Moved *.Test functionality into a single TestAccessor Class.	Matt

Section IV. Summary of Pair Development

Session #	**Time duration/place**	**Participants**	**Tasks**
1	9/25/14 3:00pm/30min	Matthew & Matt	Refactoring Node class
2	9/30/14 3:00pm/1hr	Matt & Zakk	Test GUI & mouse clicks
3	9/30/14 4:0pm/30min	Matthew & Zakk	Image Graphics
4	10/13/14 10:30am/4hrs	Matthew, Matt & Zakk	Extensive testing and bug fixing
5	10/14/14 2:50pm/1hr	Matthew, Matt & Zakk	Further Testing Create 2player bug list Start AI

6	10/20/14 11:00am/3.5hrs	Matthew, Matt & Zakk	Further Testing Fix AI bugs Fix GUI bugs

Section V. Lessons Learned

Matthew Gillatt

(1) What did you personally gain from the project?
I learned how to use the Git plug-in for Eclipse in order to organize a team project and keep track of our progress. I learned more specific usage of Git terms like pull, commit, and push. I read and comprehended code written by teammates extensively, using provided methods to complete my own, and finding bugs in them if the output is wrong. Our teamwork was certainly much cleaner and more organized than the team project I did back in CS342.

(2) What does your program do well, and what could your program do better?
The game runs quickly and cleanly, and is gives useful feedback when the user makes mistakes. An easy AI is implemented, who aims to make mills, but doesn't block or utilize flying pieces in phase 3. AI focus will be postponed to the second half of the project. The graphics are nice, but could be flashier for modern audiences.

(3) How could you improve your development process if you develop a similar game from scratch?
It would save time to set up the class hierarchy beforehand, to avoid confusing which class handles which tasks (For example, GUI isn't just graphics, but also user input like mouse clicks.) I'll use Git correctly to avoid committing the wrong files and merge correctly. If a report is needed in advance, we should have had an online document (like Google Doc) created from the start, instead of files in the project.

Matt Weaver

(1) What did you personally gain from the project?
I personally gained a lot of good experience with working with other peoples' code. One thing that really stuck out for me was working in an organized team. Most CS projects end up being "I do this, you do this, then we'll meet up in 2 weeks and put it together." In

this project, we were able to work together well and continuously. The project seemed to grow more organically, due to the frequent meetings and informal discussions.
I also gained valuable experience with testing. Since we had the tests in a separate package from the main game code, we were forced to come up with a way to access the internal variables of the various classes, while still maintaining reasonable visibility.

(2) What does your program do well, and what could your program do better?
For the most part, our program's structure seems solid. We tried to follow the single responsibility principle, which ended up making it relatively easy to track down bugs. There are a few pieces of code where the divisions of responsibility (e.g., should the AI instance be in Game, since that controls the actual game logic; or should it be it be in GUI, since that handles the interface between players and the game logic, which would make Game not have to worry about human vs AI players).
I also think the AI turned out surprisingly good, for how simple it is. It definitely could (and will) be improved, but with a fairly simple algorithm for determining moves, the AI is an unexpectedly challenging opponent. That said, it definitely could use some improvement.

(3) How could you improve your development process if you develop a similar game from scratch?
The biggest problem we encountered was a lack of planning. Had we spent more time on the initial design of the project, we probably wouldn't have had to do some of the more in-depth refactorings (like the switch from tracking what phase the game is in to just tracking what the next move should be). We also didn't do a very good job of keeping track of what we did. This lead to a lot more work, later on, as we had to look back through the Git logs and meeting notes to find things like test cases that we had implemented by hadn't explicitly written down in the backlog. Had we been doing this all along, it would have made life a lot easier.

Zakk Lambertsen

(1) What did you personally gain from the project?
I learned how agile programming is supposed to work. The frequent meetings made it easy to discuss any problems we encountered and we were able to fix them early on rather than fixing them after the fact. I also gained a better understanding of how to do JUnit testing in eclipse. Before I've only ever modified other peoples code to fit my own This is something that I haven't ever done directly. Testing GUIs is extremely difficult. I also gained further knowledge of writing Java GUIs.

(2) What does your program do well, and what could your program do better?
The program has really simple look to it. Being involved in more of the GUI work I think we could spruce up the look and feel a little bit. The AI for our game is pretty difficult to play against even though the logic behind it is really simple. We could make it have 3

different difficulty modes: easy, medium and hard.

(3) How could you improve your development process if you develop a similar game from scratch?

I think we could improve the development process of our team by meeting together more earlier on in the coding process to discuss design and structure of the game. It seemed like a frequent topic of debate was how each class was supposed to function.