Sprint 5

Jonathan Benson

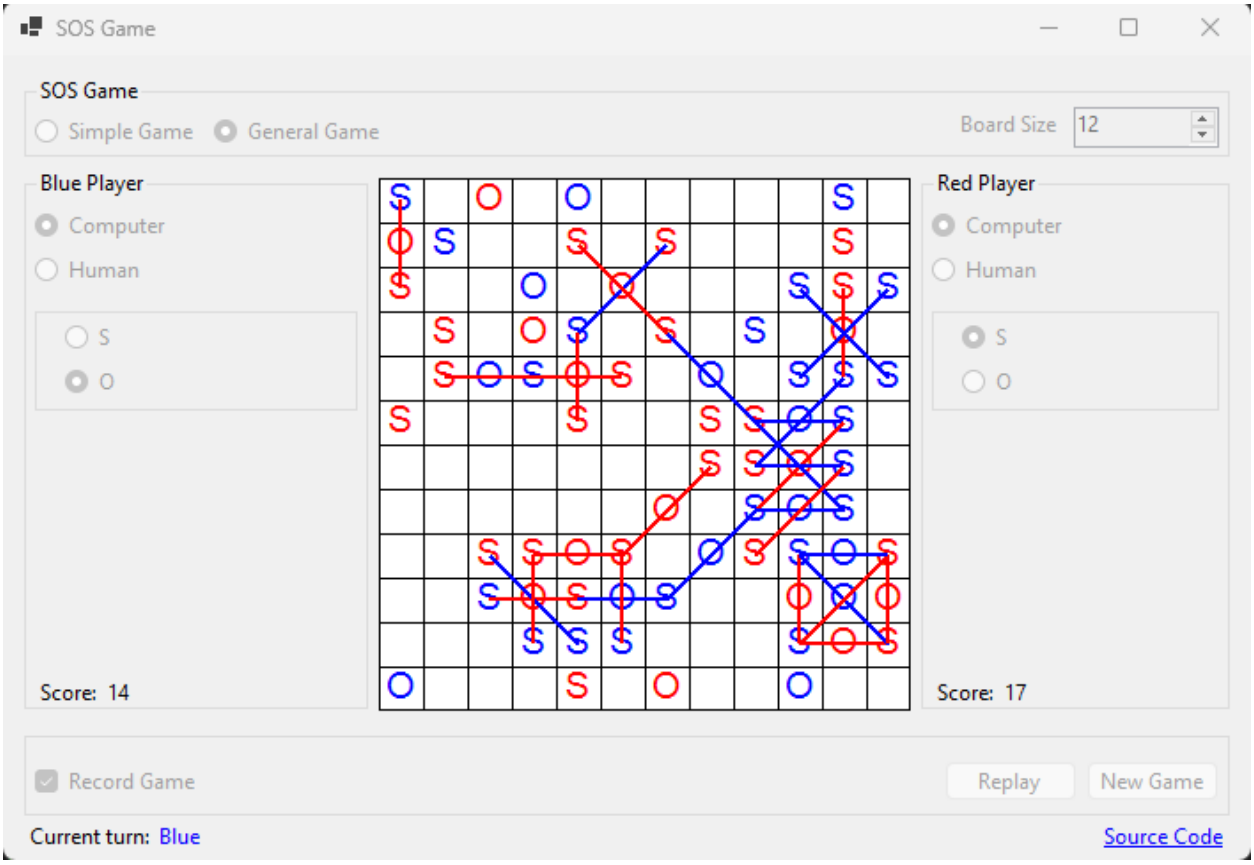
CS449

11 December 2022

GitHub Link for Source Code (sprint_5 folder): [GitHub - jonathanbenson/sosgame]

1. Demonstration Video Link: https://youtu.be/HB9f47JSXfI

Screenshot of SOS Game at the End of Sprint 5



User Stories

ID	User Story Name	User Story Description	Priority	Estimated Effort (hours)
1	Choose a board size	As a player, I want to be able to choose the board size, so that I can play SOS on different board sizes	2	1
2	Choose the game mode	As a player, I want to be able to choose the game mode, so that I can play SOS in different game modes	3	1
3	Start a new game	As a player, I want to be able to start a new game, so that I can play a new game	4	2
4	User makes a move	As a player, I want to be able to make a move, so that I can progress in the game	6	2
5	A simple game is over	As a player, I want to be able to know when a simple game is over, so that I do not waste time in the game	7	2
6	A general game is over	As a player, I want to be able to know when a general game is over, so that I do not waste time in the game	8	2
7	Game replay	As a player, I want to be able to replay the previous game, so I can analyze my performance	8	2
8	Move selection	As a player, I want to be able to select S or O, so I can play the SOS game the way it is designed to play	5	1
9	Computer makes a move	As a computer, I want to be able to make a move automatically, so I can challenge the human player	9	3

Updated Acceptance Criteria

Story ID	AC ID	AC Name	Description of Acceptance Criterion	Status
1	1.1	User attempts to select an invalid board size	Given a SOS application, when the user attempts to select an invalid board size, then the board size will stay the same	completed
	1.2	User attempts to select a board size within a game	Given a SOS application that is in a game, when the user attempts to select a board size, then the board size will stay the same	completed
	1.3	User does not select a board size	Given a SOS application, when the does not select a board size, then the board size will be set to 8	completed
	1.4	User selects a valid board size outside of a game	Given a SOS application that is not in a game, when the user selects a valid board size, then the SOS board will be rendered with the correct size and the new board size will be visible in the board size selection	completed
2	2.1	User attempts to select an invalid game mode	Given a SOS application that is not in a game, when the user attempts to select an invalid game mode, then the game mode will stay the same	completed
	2.2	User attempts to select a game mode inside of a game	Given a SOS application that is in a game, when the user attempts to select a game mode, then the game mode will stay the same	completed
	2.3	User does not select a game mode	Given a SOS application, when the user does not select a game mode, then the game mode will be set to simple game	completed
	2.4	User selects a valid game mode outside of a game	Given a SOS application that is not in a game, when the user selects a valid game mode, then the new selected game mode will be checked in the game mode selection	completed
3	3.1	User starts a new game during a game	Given a SOS application that is in a game, when the user starts a new game, then a new game mode will start with the correct game mode	completed
	3.2	User starts a new game outside of a game	Given a SOS application that is outside of a game, when the user starts a new game, then a new game will start	completed

4	4.1	User makes a move on a nonempty cell	Given an SOS application during a game, when a user makes a move on a nonempty cell, then the cell will remain unchanged and the player's turn will not change	completed
	4.2	User makes a move outside the game board	Given an SOS application during a game, when a user attempts to make a move outside of the game board, then all cells will remain unchanged and the user's turn will not change	completed
	4.3	User makes a move on an empty cell	Given SOS application during a game, when the user makes a move on an empty cell, then fill the cell with the corresponding S or O and mark an SOS on the board with the color of the current player if it completes a SOS	completed
	4.4	User makes a move not during a game	Given SOS application outside of a game, when the user makes a move, then the game state will not change	completed
	4.5	The user makes a move when it's not their turn	Given SOS application during a game, when the user makes a move and it's not their turn, their move will not count	completed
5	5.1	User makes move that wins a simple game	Given a SOS application inside a simple game, when the user makes a move that completes an SOS, then tell the user that they won the game and end the game	completed
	5.2	User makes move that does not win a simple game	Given a SOS application inside a simple game, when the user makes a move that does not complete an SOS, then switch turns between players and do not end the game	completed
	5.3	Simple game ends in a draw	Given a SOS application inside a simple, game, when the user makes a move that fills the last nonempty cell and no SOSs have been completed yet, then end the game in a draw	completed
6	6.1	User makes move that wins a general game	Given a SOS application inside a general game, when the user makes a move on the last empty square and they have more completed SOSs than their opponent, then tell the user which player won the game and end the game	completed

	6.2	User makes move that does not win a general game, but completes a SOS	Given a SOS application during a general game, when the user makes a move on an empty square that is not the last empty square and their move completes an SOS, then do not switch turns between players and do not end the game	completed
	6.3	User makes move that does not win a general game and does not complete a SOS	Given a SOS application during a general game, when the user makes a move on an empty square that is not the last empty square and their move does not complete a SOS, then switch turns between players and do not end the game	completed
	6.4	User makes move that ends general game in a draw	Given a SOS application during a general game, when the user makes a move that fills up all the squares and the number of completed SOSs by each player are the same, then tell the user that there is a draw and end the game	completed
7	7.1	User attempts to replay previous game while playing a game	Given a SOS application during a game, when the user replays the previous game, then the previous game will not be replayed	completed
	7.2	User replays game outside of a game with no previous game played	Given a SOS application outside of a game with no previous game player, when the user attempts to replay the non-existent previous game, then the non-existent previous game does not replay	competed
	7.3	User replays game outside of a game with previous game played	Given a SOS application outside of a game with a previous game played, when the user replays the previous game, then the previous game will replay from start to finish on the game board with constant time steps for each turn	competed
	7.4	User replays a game when the previous game was not recorded	Given a SOS application outside of a game with the previous game not recorded, when the user replays the previous game, then the previous game will not be replayed	
8	8.1	User attempts to select another move option besides S or O	Given an SOS application, when the user attempts to select another move option besides an S or O, then the user's move option will stay the same	completed

	8.2	User does not select a move option	Given an SOS application, when the user does not select a move option, then the user's move option will be set to "S"	completed
	8.3	User selects a move option with the values S or O	Given an SOS application, when the user selects S or O, then the user's move option will update to their desired selection	completed
	8.4	User attempts to select a move option outside of a game	Given an SOS application not during a game, when the user tries to select S or O, their move type will remain unchanged	completed
9	8.1	Computer makes a move on an empty board in a simple game or general game	Given an SOS application during a simple or general game, when there is no opportunity to complete an SOS, then make a random move on a random cell	completed
	8.2	Computer makes a move on a nonempty board in a simple game or general game when there is no opportunity to complete an SOS	Given an SOS application during a simple or general game on a nonempty board, when there is no opportunity to complete an SOS, then make a move on a random cell	completed
	8.3	Computer makes a move when there is an opportunity to complete an SOS in a simple game	Given an SOS application during a simple game, when there is an opportunity to complete an SOS, then…flip a fair coin to decide whether to either complete a SOS to win the game or to make a random move	completed
	8.4	Computer makes a move when there is an opportunity to complete only one SOS in a general game	Given an SOS application during a general game, when there is an opportunity to complete only one SOS, then flip a fair coin to decide whether to complete the SOS or make a random move	completed
	8.5	Computer makes a move when there are multiple opportunities to complete an SOS in a general game	Given an SOS application during a general game, when there are multiple opportunities to complete an SOS, then choose at random from the possible moves that complete an SOS	completed

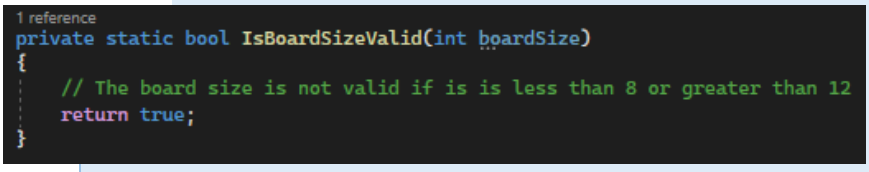
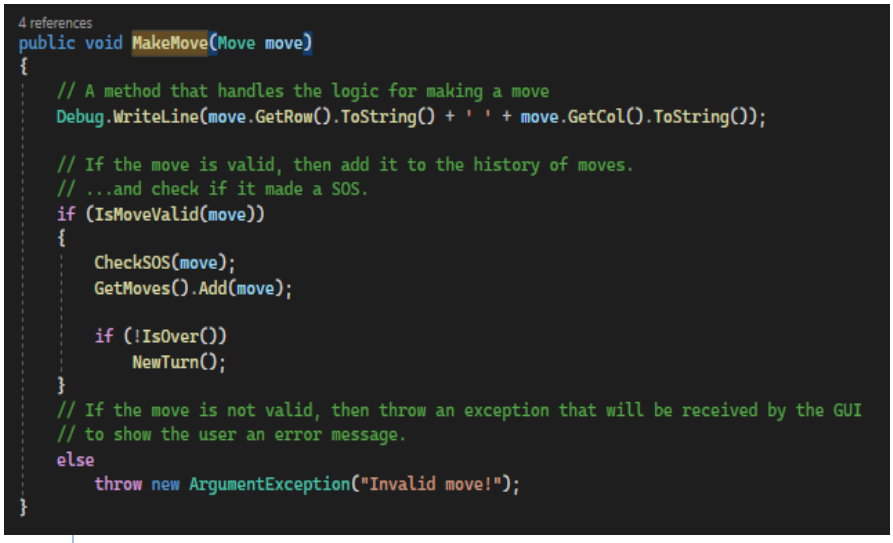
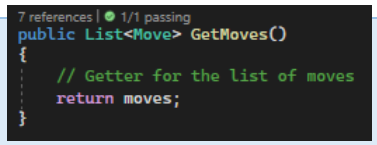
Code Review

Checklist	Checklist Item	Findings
Coding Standards	Are the naming conventions violated?	No, the naming conventions are not violated. The class and method names use Pascal case, where each word is capitalized. Additionally, the names accurately describe the purpose of the class or method. For example, Game is the base class for the game, and GetRow is a method that returns the row of a cell, and they both adhere to Pascal case.
	Is the ordering convention of method arguments violated?	No, the ordering convention of method arguments is not violated. In the code, the convention is to put any optional arguments at the end of the argument list, followed by any named arguments. For example, in the Game constructor, the optional argument recordGame is followed by the optional argument boardSize, which is followed by the optional named argument bluePlayerType.
	Any comments meaningless or inconsistent with the code?	No, the comments in the code are meaningful and consistent with the code. For example, the comment above the GameMode enumeration accurately describes the different game modes and the rules for each.
	Any code block has an inconsistent formatting style?	The code block formatting is consistent in the code above. For example, all code blocks are indented using four spaces, and all lines within code blocks are indented consistently.
	Any indentations inconsistent?	The indentation is consistent in the code above. For example, the Game class's Game constructor has its arguments indented consistently, and the Player class's MakeMove method has its code blocks indented consistently.
Design Principles	Any class/method not well-modularized?	The Game class is well-modularized in the code above. For example, the Game class has separate methods for instantiating the players, updating the game state, and checking for a winner.

	Any class with poor abstraction?	No, the classes in the code have good abstraction. For example, the Game class is an abstract base class that defines the common logic for the different types of SOS games, while the SimpleGame and GeneralGame classes are derived classes that implement the specific rules for each game mode.
	Is the visibility of any variable, method, and class inappropriate?	No, the visibility of the variables, methods, and classes in the code is appropriate. For example, the boardSize variable in the Game class is declared as private, which means it can only be accessed from within the Game class.
	Is design by contract (pre/post-condition) violated?	The design by contract (pre/post-condition) principle is not violated in the code above. For example, the Game class's MakeMove method has a precondition that checks whether the game is over before allowing the move to be made, and it has a postcondition that updates the game state after the move is made.
	Is the Open-Closed Principle violated?	Yes, the Open-Closed Principle is violated. The "Game" class contains many "if" statements that determine the behavior of the game based on the game mode. This makes the class difficult to extend and modify. The class should be designed in a way that allows new game modes to be added without modifying the existing code.
	Is the Single Responsibility Principle violated?	Yes, the Single Responsibility Principle is violated. The "Game" class contains a large number of methods and variables that relate to different aspects of the game (e.g. game logic, player management, etc.). This makes the class difficult to understand and maintain. The class should be split into smaller, more focused classes that each have a single responsibility. If I were to develop a new app, I would focus on pulling code out of larger classes into smaller classes with less responsibilities.
Code Smells	Are there magic numbers?	Yes, there are magic numbers in the code. For example, the "boardSize" variable is initialized with a value of 8, but this value is not explained or documented. Magic numbers should be avoided by using constants or named variables instead.
	Are there unnecessary global / class variable?	There are unnecessary global / class variables: The recordGame and boardSize variables are global variables that are not being used in the class and can be removed.

	Is there duplicate code?	There is duplicate code: The bluePlayer and redPlayer objects are instantiated in a similar way, with the only difference being the color of the player. This can be refactored into a single method to avoid duplication.
	Are there long methods?	There are long methods: The MakeMove method has many lines of code and can be broken down into smaller methods to improve readability.
	Is there any long parameter list?	There is a long parameter list: The Game constructor has a long list of parameters, which can make the code difficult to read and maintain.
	Is there over-complex expression?	There is an over-complex expression: The if statement in the MakeMove method is complex and can be refactored into multiple smaller if statements for readability.
	Is there switch or if-then-else that needs to be replaced with polymorphism	There is a switch or if-then-else that needs to be replaced with polymorphism: The bluePlayer and redPlayer objects are instantiated based on the PlayerType enum, which can be replaced with polymorphism to avoid the use of a switch statement.
	Any variable or method name whose intent is unclear?	Yes, the IsBoardSizeValid method name is not clear on what it does and can be renamed to something more descriptive, such as IsValidBoardSize.
	Any similar methods in different classes?	There are some similar methods in other classes. For example, the SOSEngine class also has an "EndGame" method that has different functionality from the Game.EndGame method.

Bugs

Bugs	Buggy code snippet	What is the bug?	Why is it a bug?
		The IsBoardSizeValid method checks if the board size is between 8 and 12, but it always returns true.	The IsBoardSizeValid method is intended to check if a given board size is valid, but it always returns true, so it does not actually perform this check. As a result, it will not prevent invalid board sizes from being used, which may cause unexpected behavior or errors in the program. This is a bug because the method is not fulfilling its intended purpose.
		The MakeMove method uses the Debug.WriteLine method to print a message to the console.	The Debug.WriteLine method is typically used for debugging purposes. It is used to print messages to the console during development, but it should not be used in production code because it is not intended for use by end users. Instead, production code should use a logging library or other appropriate means of outputting messages to the user. Using the Debug.WriteLine method in production code may cause unexpected behavior or performance issues.
		In the Game class, the GetMoves method returns a reference to the list of moves instead of a copy of the list.	This is a bug because it allows external code to modify the list of moves, which can cause errors in the game logic. It would be better to return a copy of

```
1 reference
private static bool IsBoardSizeValid(int boardSize)
{
    // The board size is not valid if is is less than 8 or greater than 12
    return true;
}
```

```
4 references
public void MakeMove(Move move)
{
    // A method that handles the logic for making a move
    Debug.WriteLine(move.GetRow().ToString() + ' ' + move.GetCol().ToString());

    // If the move is valid, then add it to the history of moves.
    // ...and check if it made a SOS.
    if (IsMoveValid(move))
    {
        CheckSOS(move);
        GetMoves().Add(move);

        if (!IsOver())
            NewTurn();
    }
    // If the move is not valid, then throw an exception that will be received by the GUI
    // to show the user an error message.
    else
        throw new ArgumentException("Invalid move!");
}
```

```
7 references | 1/1 passing
public List<Move> GetMoves()
{
    // Getter for the list of moves
    return moves;
}
```

	the list so that the original list cannot be modified.

Summary of Source Code

Source Code file Name	Production code or test code?	Notes	# Lines of Code
Game.cs	Production Code	SOS game base class	1073
Form1.cs	Production Code	GUI operations and control event handlers	556
BoardPainter.cs	Production Code	Drawing board items (grid lines, S's & O's, SOS's)	215
SOSEngine.cs	Production Code	High-level SOS application logic (starting games, etc.)	209
AccessibilityManager.cs	Production Code	Manage accessibility and visilibility of GUI controls	153
ComputerPlayer.cs	Production Code	Computer player logic	116
Move.cs	Production Code	Logic associated with a move	87
Player.cs	Production Code	SOS player base class	66
GeneralGame.cs	Production Code	SOS game logic associated with a general game	53
SOSLine.cs	Production Code	Simple class model for a completed SOS line	53
SimpleGame.cs	Production Code	SOS game logic associated with a simple game	43
Replay.cs	Production Code	Replay class that associated with the playing of replays	42
HumanPlayer.cs	Production Code	Human player logic	38
GameTest.cs	Test Code	Testing for the Game class	431
GeneralGameTest.cs	Test Code	Testing of the GeneralGame class	201
ComputerPlayerTest.cs	Test Code	Testing for the ComputerPlayer class	161
SimpleGameTest.cs	Test Code	Testing of the SimpleGame class	137
AccessibilityManagerTest.cs	Test Code	Testing for the AccessibilityManagerClass	120
SOSEngineTest.cs	Test Code	Testing for the SOSEngine class	88
ReplayTest.cs	Test Code	Testing for the Replay class	84
PlayerTest.cs	Test Code	Testing for the Player class	47
MoveTest.cs	Test Code	Testing for the Move class	41
HumanPlayerTest.cs	Test Code	Testing for the HumanPlayer class	32
		Total	**4046**

Lessons Learned

As a college computer science student, the project of building an application to play the SOS game was a valuable learning experience. Not only did it allow me to put into practice the software engineering principles that I had learned in my studies, such as object-oriented design, refactoring, and testing, but it also gave me the opportunity to experiment with different technologies and frameworks, such as C#, .NET, WinForms, and the MS Test unit testing framework.

One of the things that I personally gained from the project was a better understanding of the Scrum process and how it can be used to manage the development of a software project. By working in sprints and using the agile methodology, I was able to break down the project into manageable chunks and focus on delivering functionalities incrementally. This helped me to stay organized and on track and allowed me to make progress on the project in a structured and efficient way.

Another thing that I gained from the project was the experience of building a complete application from scratch. This allowed me to see the whole development process, from initial planning and design to final testing and deployment. This was a valuable learning experience, as it allowed me to see how the different stages of development fit together and how they contribute to the overall quality of the application.

Overall, I believe that my project does well in terms of functionality. The application can play the SOS game according to the rules, and it provides a user-friendly interface that allows users to play the game easily. The application also makes use of object-oriented design principles, which helps to make the code modular and reusable.

One area where the project could do better is in terms of testing. While the project does make use of the MS Test framework for unit testing, the coverage of the tests is not as complete as it could be. There are some areas of the code that are not adequately tested, which could lead to bugs and other issues in the future.

If I were to develop a similar game from scratch, I would focus on improving my testing process. I would make sure to write comprehensive unit tests that cover all aspects of the code, and I would also consider using other testing techniques, such as integration testing and more comprehensive user acceptance testing, to ensure that the application is as bug-free as possible.

I would also try to incorporate feedback from users into the development process. By collecting feedback from users and incorporating it into the design and implementation of the application, I could ensure that the application meets the needs and expectations of the users. This would help to improve the overall quality of the application and make it more user-friendly.

Overall, the project of building an application to play the SOS game was a genuinely enjoyable experience. It allowed me to put into practice most of the things I've picked up over the years in my computer science classes, and it gave me the opportunity to experiment with different technologies and frameworks in which I was personally interested. By focusing on improving my testing process and incorporating user feedback into the development process, I believe that I could improve the development process and create a better game from scratch.