**Project Sprint #2**

Implement the following features of the SOS game: (1) the basic components for the game options (board size and game mode) and initial game, and (2) S/O placement for human players ***without*** checking for the formation of SOS or determining the winner. The following is a sample interface. The implementation of a GUI is strongly encouraged. You should practice object-oriented programming, making your code easy to extend. It is important to separate the user interface code and the game logic code into different classes (refer to the TicTacToe example). xUnit tests are required.

|  |  |  |
| --- | --- | --- |
| SOS Icon  Description automatically generated Simple game Icon  Description automatically generated General game Board size  8 | | |
| Blue player  Icon  Description automatically generated S  Icon  Description automatically generated O | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  | | O |  |  |  |  |  |  |  | |  |  | S | O | S |  |  |  | |  |  |  |  | S |  |  |  | |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | S | | Red player  Icon  Description automatically generated S  Icon  Description automatically generated O |
|  | Current turn: blue (or red) |  |

Figure 1. Sample GUI layout of the Sprint 2 program

**Deliverables:**

1. **Demonstration (8 points)**

Submit a video of no more than three minutes, clearly demonstrating that you have implemented the required features and written some automated unit tests. In the video, you must explain what is being demonstrated.

|  |  |  |
| --- | --- | --- |
|  | **Feature** |  |
| 1 | Choose board size |  |
| 2 | Choose game mode |  |
| 3 | Initial game of the chosen board size and game mode |  |
| 4 | “S” moves |  |
| 5 | “O” moves |  |
| 6 | Automated unit tests |  |
| … |  |  |

1. **Summary of Source Code (1 points)**

|  |  |  |
| --- | --- | --- |
| Source code file name | Production code or test code? | # lines of code |
| sosGame.h | Production code | 40 |
| sosGUI.h | Production code | 75 |
| main.cpp | Production code | 11 |
| sosGames.cpp | Production code | 114 |
| sosGUI.cpp | Production code | 289 |
| tst\_testemptyboard.cpp | Test code | 72 |
| tst\_testsosgeneralgame.cpp | Test code | 84 |
| tst\_testsosgui.cpp | Test code | 117 |
| tst\_testsossimplegame.cpp | Test code | 85 |
| Total | | 887 |

**You must submit all source code to get any credit for this assignment.**

1. **Production Code vs User stories/Acceptance Criteria (3 points)**

Update your user stories and acceptance criteria from the previous assignment and ensure they adequately capture the requirements. Summarize how each of the following user story/acceptance criteria is implemented in your production code (class name and method name etc.)

|  |  |
| --- | --- |
| **User Story ID** | **User Story Name** |
| 1 | Choose a board size |
| 2 | Choose the game mode of a chosen board |
| 3 | Start a new game of the chosen board size and game mode |
| 4 | Make a move in a simple game |
| 6 | Make a move in a general game |

|  |  |  |  |
| --- | --- | --- | --- |
| **User Story ID and Name** | **AC**  **ID** | **Description of Acceptance Criterion** | **Status (completed, toDo, inProgress)** |
| 1 Choose Game Mode | 1.1 | AC 1.1 <Option Game Mode: Simple>  Given User selects game mode  When User selects “Simple Game” option  Then the System Starts a new game under the mode Simple with a clear game from scratch. | Complete |
| 1.2 | AC 1.2 <Option Game Mode: General>  Given User selects game mode  When User Selects “General Game” option  Then the System Starts a new game under the mode General with a clear game from scratch. | Complete |
| 2 Choose the Board Size | 2.1 | AC 2.1<Board Size>  Given User selects the board size  When User enters “Board Size”  Then the System starts with an adjusted size of the board for the game to match the value of the board size. | Complete |
| 3 Setup New Game | 3.1 | AC 3.1 <New Game >  Given User in middle of Game or Game Over, wants new game  When User selects the “New Game” option  Then the system starts clears the board and has user to select a game mode. | Complete |
| 4 Make a Move Simple Game | 4.1 | AC 4.1 <User wants to place an S or O on the board >  Given User is in a current game and it is there turn  When User Selects an empty square and places an S or an O  Then System adds an S or an O onto the empty square the user selected and inputs. Next Users turn. | Complete |
|  | 4.2 | AC 4.2 <User wants to place an S or O on the board and selects invalid placement outside of board or tries to place on occupied cell>  Given User is in a current game and it is their turn, if user selects a position outside the board or on a cell that is already occupied, then nothing will happen and it will remain same User’s turn. | Complete |
| 5 Make a Move General Game | 5.1 | AC 5.1 <User wants to place an S or O on the board >  Given User is in a current game and it is there turn  When User Selects an empty square and places an S or an O  Then System adds an S or an O onto the empty square the user selected and inputs. If SOS created, keep users turn, else, next Users turn. | Partially Complete |
| 5.2 | AC 5.2 <User wants to place an S or O on the board and selects invalid placement outside of board or tries to place on occupied cell>  Given User is in a current game and it is their turn, if user selects a position outside the board or on a cell that is already occupied, then nothing will happen and it will remain same User’s turn. | Complete |
|
| 6 Determine if Simple Game is Over | 6.1 | AC 6.1 <Check Game Over: Simple>  Given User is in the middle of a Simple Game  When A User has an S-O-S connected together  Then the System ends the game and determines winner. If same number, then it’s a draw. | toDo |
| 7 Determine if General Game is Over | 7.1 | AC 7.1 <Check Game Over: General>  Given User is in the middle of a General Game  When all the squares are filled.  Then the System ends the game and determines the winner based off who has the most S-O-S’s connected. If same number, then it’s a draw. | toDo |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Story ID and Name** | **AC ID** | **Class Name(s)** | **Method Name(s)** | **Status (complete or not)** | **Summary Code** |
| 1 Choose Game Mode | 1.1 | sosGame,  sosGUI | void newGame(int boardSize, int gameMode);  void setBoardMode(int boardMode);  void newSOSGame(sosGame newGame);  void setGameStart();  void drawGameGrid();  void drawGameBoard(); | Complete | sosGUI class creates a default GUI for SOS Game and calls sosGame class to create an SOS Game that hold the game information. drawGameGrid() draw the SOS Game board and drawGameBoard draws the current SOS Game board with the current state of the game. On the GUI, when a Game Mode of Simple is selected, the sosGame within sosGUI class is updated to that game mode using the method call setBoardMode or newGame which takes an input of a board size and game mode and creates a new game and sets GUI elements to match and display it. setGameStart sets the GUI elements on start of a new Game. |
|  | 1.2 | sosGame,  sosGUI | void newGame(int boardSize, int gameMode);  void setBoardMode(int boardMode);  void newSOSGame(sosGame newGame);  void setGameStart();  void drawGameGrid();  void drawGameBoard(); | Complete | sosGUI class creates a default GUI for SOS Game and calls sosGame class to create an SOS Game that hold the game information. drawGameGrid() draw the SOS Game board and drawGameBoard draws the current SOS Game board with the current state of the game. On the GUI, when a Game Mode of General is selected, the sosGame within sosGUI class is updated to that game mode using the method call setBoardMode or newGame which takes an input of a board size and game mode and creates a new game and sets GUI elements to match and display it. setGameStart sets the GUI elements on start of a new Game. |
| 2 Choose the Board Size | 2.1 | sosGame,  sosGUI | void initGameBoard();  void newGame(int boardSize, int gameMode);  void setBoardSize(int boardSize);  void newSOSGame(sosGame newGame);  void setGameStart();  void drawGameGrid();  void drawGameBoard();  void onBoardSizeSpinBoxValueChange(int); | Complete | sosGUI class creates a default GUI for SOS Game and calls sosGame class to create an SOS Game that hold the game information. drawGameGrid() draw the SOS Game board and drawGameBoard draws the current SOS Game board with the current state of the game. On the GUI, when a Board Size of the Spin Box is chosen, it calls setBoardSize to update it for the game logic class or can create a new game instantly as well using new Game to set board size and game mode. Whenever the value of the spinbx for board size is changed, it updates the drawGameGrid to change to the size of the board. setGameStart sets the elements for the new game of game board GUI when a new game of board size and mode is made. |
| 3 Setup New Game | 3.1 | sosGame,  sosGUI | int getRows();  int getCols();int getGameState();  int getGameMode();  int getTurn();  int getCell(int row, int col);  void initGameBoard();  void updateStatus();  void setGameStart();  void newSOSGame(sosGame newGame);  void mousePressEvent(QMouseEvent \*event);  void drawGameGrid();  void drawGameBoard(); | Complete | sosGUI class creates a default GUI for SOS Game and calls sosGame class to create an SOS Game that hold the game information. drawGameGrid() draw the SOS Game board and drawGameBoard draws the current SOS Game board with the current state of the game. When setting up a new game to be called with newSOSGame being applied to the sosGUI class, it calls getRows, getCols, getGameMode, getTurn, and getCell in order to get the sosGame information and to update the GUI layout to match the current state of the game. On mouePressEvent, checks for validation of an SOS move within a current game and valid cell. |
| 4 Make a Move Simple Game | 4.1 | sosGame,  sosGUI | int getRows();  int getCols();  int getGameState();  int getGameMode();  int getTurn();  int getCell(int row, int col);  void makeMove(int row, int col, int letter);  void updateStatus();  int getPlayerTurnLetter();  void mousePressEvent(QMouseEvent \*event);  void drawGameGrid();  void drawGameBoard(); | Complete | sosGUI class creates a default GUI for SOS Game and calls sosGame class to create an SOS Game that hold the game information. drawGameGrid() draw the SOS Game board and drawGameBoard draws the current SOS Game board with the current state of the game. When setting up a new game to be called with newSOSGame being applied to the sosGUI class, it calls getRows, getCols, getGameMode, getTurn, and getCell in order to get the sosGame information and to update the GUI layout to match the current state of the game. On mouePressEvent, checks for validation of an SOS move within a current game and valid cell. If it is valid move in a playing game, makeMove is called to check for the move and if the cell is empty or not. As long as a valid move, then the move is made and updated to the game. This then in turn updates the game board GUI with drawGameGrid and drawGameBoard for the Simple Game as long as it is a valid cell and is empty, then updates status of the GUI game. |
|  | 4.2 | sosGame,  sosGUI | int getRows();  int getCols();  int getGameState();  int getGameMode();  int getTurn();  int getCell(int row, int col);  void makeMove(int row, int col, int letter);  int getPlayerTurnLetter();  void mousePressEvent(QMouseEvent \*event);  void drawGameGrid();  void drawGameBoard(); | Complete | sosGUI class creates a default GUI for SOS Game and calls sosGame class to create an SOS Game that hold the game information. drawGameGrid() draw the SOS Game board and drawGameBoard draws the current SOS Game board with the current state of the game. When setting up a new game to be called with newSOSGame being applied to the sosGUI class, it calls getRows, getCols, getGameMode, getTurn, and getCell in order to get the sosGame information and to update the GUI layout to match the current state of the game. On mouePressEvent, checks for validation of an SOS move within a current game and valid cell. If it is valid move in a playing game, makeMove is called to check for the move and if the cell is empty or not. This then in turn updates the game board GUI with drawGameGrid and drawGameBoard for the Simple Game as long as it is a valid cell and is empty, then updates status of the GUI game to the other players turn to make a move. If it is an invalid move of the Simple Game, nothing happens and it continues to be the same users turn. |
| 5 Make a Move General Game | 5.1 | sosGame,  sosGUI | int getRows();  int getCols();  int getGameState();  int getGameMode();  int getTurn();void makeMove(int row, int col, int letter);  int getCell(int row, int col);  void updateStatus();  int getPlayerTurnLetter();  void mousePressEvent(QMouseEvent \*event);  void drawGameGrid();  void drawGameBoard(); | Partially Complete | sosGUI class creates a default GUI for SOS Game and calls sosGame class to create an SOS Game that hold the game information. drawGameGrid() draw the SOS Game board and drawGameBoard draws the current SOS Game board with the current state of the game. When setting up a new game to be called with newSOSGame being applied to the sosGUI class, it calls getRows, getCols, getGameMode, getTurn, and getCell in order to get the sosGame information and to update the GUI layout to match the current state of the game. On mouePressEvent, checks for validation of an SOS move within a current game and valid cell. If it is valid move in a playing game, makeMove is called to check for the move and if the cell is empty or not. As long as a valid move, then the move is made and updated to the game. This then in turn updates the game board GUI with drawGameGrid and drawGameBoard for the Simple Game as long as it is a valid cell and is empty, then updates status of the GUI game. |
|  | 5.2 | sosGame,  sosGUI | int getRows();  int getCols();  int getGameState();  int getGameMode();  int getTurn();void makeMove(int row, int col, int letter);  int getCell(int row, int col);  int getPlayerTurnLetter();  void mousePressEvent(QMouseEvent \*event);  void drawGameGrid();  void drawGameBoard(); | Complete | sosGUI class creates a default GUI for SOS Game and calls sosGame class to create an SOS Game that hold the game information. drawGameGrid() draw the SOS Game board and drawGameBoard draws the current SOS Game board with the current state of the game. When setting up a new game to be called with newSOSGame being applied to the sosGUI class, it calls getRows, getCols, getGameMode, getTurn, and getCell in order to get the sosGame information and to update the GUI layout to match the current state of the game. On mouePressEvent, checks for validation of an SOS move within a current game and valid cell. If it is an invalid move of the General Game, nothing happens and it continues to be the same users turn. |
| 6 Determine if Simple Game is Over | 6.1 | Not Complete | Not Complete | Not Complete |  |
| 7 Determine if General Game is Over | 7.1 | Not Complete | Not Complete | Not Complete |  |

1. **Tests vs User stories/Acceptance Criteria (3 points)**

Summarize how each of the user story/acceptance criteria is tested by your test code (class name and method name) or manually performed tests.

|  |  |
| --- | --- |
| **User Story ID** | **User Story Name** |
| 1 | Choose a board size |
| 2 | Choose the game mode of a chosen board |
| 3 | Start a new game of the chosen board size and game mode |
| 4 | Make a move in a simple game |
| 6 | Make a move in a general game |

4.1 Automated tests directly corresponding to the acceptance criteria of the above user stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Story ID and Name** | **Acceptance Criterion ID** | **Class Name (s) of the Test Code** | **Method Name(s) of the Test Code** | **Description of the Test Case (input & expected output)** |
| 1 Choose Game Mode | 1.1 | TestSOSSimpleGame | testEmptySimpleGame() | Test case takes a new game input of a board size and an input of simple game mode. It compares that the game mode is a simple game mode and compares every cell of the game to make sure a new simple game of all empty cells were created. Output pass as long as all cells are empty of a new game. |
|  | 1.2 | TestSOSGeneralGame | testEmptyGeneralGame() | Test case takes a new game input of a board size and an input of general game mode. It compares that the game mode is a general game mode and compares every cell of the game to make sure a new general game of all empty cells were created. Output pass as long as all cells are empty of a new game. |
| 2 Choose the Board Size | 2.1 | TestSOSGUI | testBoardSizeGUI() | Test case takes an input for a board size. Output passes as long as it compares that the board size input matches the board size of the game board. |
| 3 Setup New Game | 3.1 | TestSOSGUI | testNewGameBoardGUI() | Test case takes an input for a board size and a game mode to create a new game board. Output passes as long as the input board size and input game mode matches the current game board and size, and checks to make sure all of the cells of the new board size are empty. |
| 4 Make a Move Simple Game | 4.1 | TestSOSSimpleGame | testEmptySimpleGame(),  testMoveSimpleGame() | Test case takes a new game input of a board size and an input of simple game mode. It compares that the game mode is a simple game mode and compares every cell of the game to make sure a new simple game of all empty cells were created. Output pass as long as all cells are empty of a new game.  Test case take an input for a new simple game and input move for a Letter S at one location and a Letter O at another location. Output passes as long as compares that game mode is simple, and that there is a Letter S and Letter O in corresponding locations. |
|  | 4.2 | TestSOSSimpleGame | testInvalidRowSimpleGame(), testInvalidColumnSimpleGame(), | Test case compares an input cell location outside of the of valid rows size and outputs as passed as long as it matches as null.  Test case compares an input cell location outside of the valid columns size and outputs as passed as long as it matches as null. |
| 5 Make a Move General Game | 5.1 | TestSOSGeneralGame | testEmptyGeneralGame(),  testMoveGeneralGame() | Test case takes a new game input of a board size and an input of general game mode. It compares that the game mode is a general game mode and compares every cell of the game to make sure a new general game of all empty cells were created. Output pass as long as all cells are empty of a new game.  Test case take an input for a new general game and input move for a Letter S at one location and a Letter O at another location. Output passes as long as compares that game mode is general, and that there is a Letter S and Letter O in corresponding locations. |
|  | 5.2 | TestSOSGeneralGame | testInvalidRowGeneralGame(), testInvalidColumnGeneralGame(), | Test case compares an input cell location outside of the of valid rows size and outputs as passed as long as it matches as null.  Test case compares an input cell location outside of the valid columns size and outputs as passed as long as it matches as null. |
| 6 Determine if Simple Game is Over | 6.1 | Not Complete | Not Complete | Not Complete |
| 7 Determine if General Game is Over | 7.1 | Not Complete | Not Complete | Not Complete |

4.2 Manual tests directly corresponding to the acceptance criteria of the above user stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Story ID and Name** | **Acceptance Criterion ID** | **Test Case Input** | **Test Oracle (Expected Output)** | **Notes** |
| 1 Choose Game Mode | 1.1 | Select Simple Game. | Sets Game Mode to selected Simple Game |  |
|  | 1.2 | Select General Game | Sets Game Mode to selected General Game |  |
| 2 Choose Board Size | 2.1 | Select Board size on Spin Box for Board Size | Sets Board Size to selected board size |  |
| 3 Setup New Game | 3.1 | Sets up a new game with selected board size and selected game mode | New Game started, board size adjusted, board mode set to chosen options |  |
| 4 Make a Move Simple Game | 4.1 | Place an S and O for each users within a valid empty cell for a simple game | S and O are placed within the selected empty cells |  |
|  | 4.2 | Make a move outside valid board size cells of Simple Game | Nothing happens, remains current users turn |  |
| 5 Make a Move General Game | 5.1 | Place an S and O for each users within a valid empty cell for a general game | S and O are placed within the selected empty cells |  |
|  | 5.2 | Make a move outside valid board size cells of General Game | Nothing happens, remains current users turn |  |

4.3 Other automated or manual tests not corresponding to the acceptance criteria of the above user stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Test Input** | **Expected Result** | **Class Name of the Test Code** | **Method Name of the Test Code** |
|  |  |  |  |  |
|  |  |  |  |  |