Sprint 3

Jordan Taranto
CS449
Sprint-2
https://github.com/Jordinaa/cs449/tree/main/sprint3

1. Demonstration (9 points)

Submit a video of no more than five minutes, clearly demonstrating the following features.
(a) A simple game that the blue player is the winner
(b) A simple draw game with the same board size as (a)
(c) A general game that the red player is the winner, and the board size is different from (a)
(d) A general draw game with the same board size as (c)
(e) Some automated unit tests for the simple game mode
(f) Some automated unit tests for the general game mode

In the video, you must explain what is being demonstrated.

2. Summary of Source Code (1 points)

https://github.com/Jordinaa/cs449/tree/main/sprint3

Source code file name	Production or Test Code?	# Lines of Code
GUI.py	Production	145
test_GUI.py	Test	73
main.py	Production	9
Total		227

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

User Story ID and Name	AC ID	Class Name(s)	Method Name(s)	Status (complete or not)	Notes (optional)
1	1.1	`GUI`	`self.board_size = tk.IntVar(value=5)`	Complete	
2	2.1	`GUI`	`self.game_type_frame = tk.Frame(self.top_frame)`	Complete	Dynamic board added
	2.2	`GUI`	`def __init__(self):`	Complete	GUI initialized in class
3	3.1	`GUI`	`start_new_game(self)`	Complete	
4	4.1	`GUI`	`handle_grid_click(self, row, col)`	Complete	

User Story ID and Name	AC ID	Class Name(s)	Method Name(s)	Status (complete or not)	Notes (optional)
5	5.1	`Logic`	`def check_for_sos(self, row, col):`	Complete	
6	6.1	`GUI`	`def __init__(self):`	Complete	GUI initialized in class
7	7.1	`Logic`	`def handle_grid_click(self, row, col):`	Complete	Updated handled grid click due to logic issue

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

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
5	Make a move in a general game
6	Make SOS in simple game
7	Make SOS in general game

4.1 Automated tests directly corresponding to some acceptance criteria

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	1.1	`TestGUI`	None	None
2	2.1	`TestGUI`	`test_inital_state`	None
	2.2	`TestGUI`		
3	3.1	`TestGUI`	`test_inital_state`	Checks initial state of application it verifies that the current turn is set to 'blue' and default board size is 5
4	4.1	`TestGUI`	`test_handle_grid_click`	- updates the cells to the currently selected letter in my case 'S' - Change text color to match current player - switches turns between players after each click on the grid
6	6.1	`TestGUI`	`def test_simple_game_winner(self):`	Simulates moves to form and SOS horizontally

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)
7	7.1	`TestGUI`	`def test_general_game_scoring(self):`	Simulates moves to form an SOS horizontally and fill up the board

4.2 Manual tests directly corresponding to some acceptance criteria

- **Video - for manual test**

4.3 Other automated or manual tests not corresponding to the acceptance criteria

Number	Test Input	Expected Result	Class Name of the Test Code	Method Name of the Test Code

5. Describe how the class hierarchy in your design deals with the common and different requirements of the Simple Game and the General Game? (4 points)

Simple game and General games are both managed under the GUI class. This class takes care of everything from setting up the game board the handling player actions. Depending on whether the game is set to Simple or General, the same functions decide how the game reacts like when someone wins. This setup keeps everything pretty straightforward since there is no need for separate classes for each game mode, but it also means that all the differences between the game modes are mixed together in the same set of instructions. this is ok for this game since it just has two modes but if it had more it would make it much more complex. So moving forward splitting the game modes into their own classes and using inheritance in the GUI class or composition depending on the games requirements.