Project Sprint #5

Video Demonstration
https://www.loom.com/share/7f4f401645e74a85bec9dd08b91db8fe?sid=9b1f6b4a-a0e9-4ad6-a734-4ef3b0f31a62

The main tasks of this assignment are:

(1) Adding the feature of recording a game into a text file. The user story and acceptance criteria of both record and replay are required, but the implementation of replay is for extra credit (up to 2 points in the weighted total).
(2) Conducting a code review exercise.
(3) Summarizing the lessons learned from Sprint 0 through Sprint 5.

The following is a sample GUI layout of the final product, where "Replay" is optional.

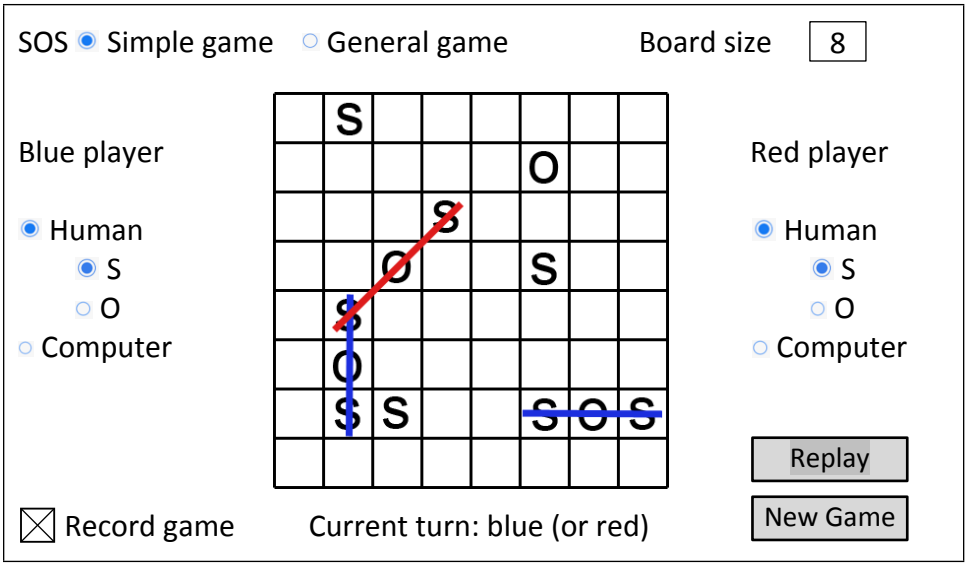

Figure 1. Sample GUI layout of the final product

Total points: 16

1. Demonstration (6 points)

Submit a video of no more than 8 minutes, clearly demonstrating that you have implemented all the features in the following table. In the video, you must explain what is being demonstrated.

	Feature
1	A complete simple game of two human players is recorded
2	A complete general game of two human players is recorded
3	A complete simple game of human-computer players is recorded
4	A complete general game of human-computer players is recorded
5	A complete simple game of computer-computer players is recorded
6	A complete general game of computer-computer players is recorded

If you have implemented the "replay" feature for extra credit, you should include its demonstration in the video.

2. User Stories and Acceptance Criteria for the Record/Replay Requirements (1 points)

- **User Story Template**: As a <role>, I want <goal> [so that <benefit>]

Add or delete rows as needed.

ID	User Story Name	User Story Description	Priority	Estimated effort (hours)
20	Record moves to text file.	As a SOS player I want to have an option to record the moves of the previous game, so that I can revise my moves and get better.	1	2

User Story ID and Name	AC ID	Description of Acceptance Criterion	Status (completed, toDo, inPprogress)
20 Record moves to text file.	20.1	AC 8.1 When the user finished the game, he should be able to see the moves that he did in that game. Given the user finished his game. When the new game starts. Then provide the user with the moves of the previous game.	Completed

3. Code Review (2 points)

Apply source code review to one or two most important classes (and other classes if time permits) and report the findings. In addition to looking for bugs, the review should check: (1) whether the entire project has followed the coding standard in a consistent manner, (2) whether the project has followed the design principles introduced in class, and (3) whether there are code smells that indicate the need for refactoring. The following checklists provide basic guidelines. You may add new items to each of the checklists.

Make sure your answers resulted from the code review exercise. If there is no finding for an entry, you should provide an explanation. For example, if your answer to "Are the naming conventions violated?" is no, you should describe a naming convention and present an example. You will receive no credit for this part if your answers are simply yes or no without additional information.

Classes that have been reviewed:
Date/time duration of the code review exercise:

Checklist	Checklist Item	Findings
Coding Standards	Are the naming conventions violated?	No, I named all my variables, so it was easy to understand what they do. For example self.trackMoves list is used to track moves of the players. And self.current_player is a counter so computer knows which player turn is it: First or Second
	Is the ordering convention of method arguments violated?	I do not think that the ordering convention of method arguments violated. I do not have more than 2 arguments for each function. In this case tk.Button(self.root, text="Make Move", command=self.make_move_button_clicked) the argument are in the right order.
	Any comments meaningless or inconsistent with the code?	No, I kept me code clear. Any comments that are not needed to understand the logic of my code were deleted.
	Any code block has an inconsistent formatting style?	No, during the project I tried to apply an object-oriented design and follow it.
	Any indentations inconsistent?	No inconsistent indentations observed.
	…	
Design Principles	Any class/method not well-modularized?	I have 2 classes in my code. SOSGameGUI and getInput. I think they modalized pretty well. The SOSGameGUI has a lot of functions for implimanting the game, while getInput has only 2 functions since it is only used for getting the input that characterizes the game.

	Any class with poor abstraction?		
	Is the visibility of any variable, method, and class inappropriate?	In my game every method and class has a public visability ,so everybody has an excess to them. I do not remember any assignment mentioning that the visibility has to be specific.	
	Is design by contract (pre/post-condition) violated?	The only post condition that could be violated, is in the general game when the computer makes the last move it is for some reason do not show on board where it placed a token, even though the program is working correct and the last token was played by computer.(It just does not show it on the grid of squares)	
	Is the Open-Closed Principle violated?	Yes. Every time I started to do the new sprint I had to change the Old code to make a new code run. As A beginner I is hard for me to make a perfect code for every sprint.	
	Is the Single Responsibility Principle violated?	No, each class seems to have a single responsibility.	
Code Smells	Are there magic numbers?	No magic numbers are observed.	
	Are there unnecessary global / class variable?	No unnecessary global or class variables are observed.	
	Is there duplicate code?	No duplicate code is observed. I am just using similar functions clear_file_ first() and clear_file_second() in order to erase the context of both of text files, related to printing the recorded moves. Also there is a duplicate code appears in the simple_game and general_game functions. self.output_text.delete('1.0', tk.END) self.clear_file_second() self.list_for_review_second.clear() self.move_text_file() self.list_for_review.clear() self.clear_file_first() self.file2_window() The code manages the recoding game and displaying the previous game.	
	Are there long methods?	The SOSGameGUI class contains some methods with multiple responsibilities, such as simple_game and general_game, which might be considered long.	
	Is there any long parameter list?	No, there are no methods with excessively long parameter lists.	
	Is there over-complex expression?	The check_sos1 method in the SOSGameGUI class contains complex logic for checking SOS formations. While it's not necessarily over-complex, it could be refactored for clarity.	
	Is there switch or if-then-else that needs to be replaced with polymorphism	Polymorphism could be added in the perform_AI_move function.	
	Any variable or method name whose intent is unclear?	No variable or method names with unclear intent are observed.	
	Any similar methods in different classes?	No, I have only 2 classes in the main file	
	…		
Bugs	**Buggy code snippet**	**What is the bug?**	**Why is it a bug?**
	The program does not like when user clicks on the make_move button without placing a symbol on the grid. But it does not result in error. I could not fing more bugs, but It does not mean that my program does not have them.		

4. Summary of All Source Code (1 points)

Source code file name	Production code or test code?	# lines of code
sprint5.py	Production code	387
test_sprint5.py	Test code	112
	Total lines of code	499

You will receive no credit for this assignment unless your complete source code is submitted.

5. Summarize the lessons learned from the entire project by answering the following questions from the perspectives of development processes, coding, design, refactoring, and testing **(6 points)**:

- What did you personally gain from the project?
- What does your project do well, and what could your project do better?
- How could you improve your development process if you develop a similar game from scratch?

Minimum requirement for (5): One full page single spaced, font size no bigger than 12 points.

SOS Project Reflection

The project was an incredible opportunity for me to learn how the code's design and actual writing works. Before this class, this was the first time I had done such complex and lengthy projects. I am sure this work will greatly help me and benefit my future career. Now, not only do I have a good project to put on my resume, but I also have basic experience working on a sprint-based type of project.

After three months of working on the SOS game, my way of writing code changed a lot. The sprint-based way of work taught me efficient ways of working with object-oriented design. It also helped me to always think about the future steps. For example, in the beginning, after sprint 2, when the first code was written, on sprint 3, I had to sit for hours and understand again the code that I wrote. Towards the end of the semester, I learned to always write a code in that is modulized and have room for improvement, so that if I need an extra counter in the __init__ function, I do not need to rewrite have of my class functions. Another aspect that I definitely improved, is my use of AI skills. I was able to evolve my debugging process and successfully communicate with Large Language Models. I also learned how to always demand more from myself and go beyond the limits. On the last sprint, the implementation of Visual of the replay of the previous game was not required. It was for the extra credit. Even though I was tired and had a lot other exams to study, I was able to successfully complete the replay option for my game.

First of all, the project was successful overall. The worst part was building an AI bot to play with. I spent many frustrating hours on this sprint in the library. In my opinion, I could have used more classes in my code and used inheritance. Also, until the last sprint, when I ran a program with a general game and when the last move is by the AI bot, for some reason, the symbol was not shown on the grid, even though the program knew what token was placed and it got evaluated for the determination of the winner. I am not sure how, but during the last sprint this problem resolved itself.

As I said earlier in the previous paragraph, if I developed a similar game from scratch, I would start with more classes. This way, the code would be more logical and understandable. Even though my final code is not that long, about 350 lines, I still think there is a way to do the whole program in about 200 lines or maybe even less.

The project was a great challenge for me and caused many hours of frustration from debugging the code. At the same time, I learned a lot about managing coding projects and about time management overall. I also can not mention that my LLM skills grew significantly compared to the one I had before this semester.