Project 1: Build a system - Team’s choice of language and platform

Due Date: 11:59 PM, Sunday, September 15, 2024

- All teams must have a code freeze as of the due date.
- The timestamp will be judged by the final commit on your master branch of the team's GitHub repository.
- You can continue to work on other branches but cannot update your master branch after the freeze date.
- You must demo your code during your GTA/Team meeting based on that master branch as of the code freeze.
- All artifacts (code and documentation) must be in your repository on your master branch.
- Your Team Peer Reviews for each team member are also due at this time in Canvas.

Overview

- Let's make Battleship!
- Battleship is a two-player game.
- Both players secretly place 1 to 5 ships on a 10x10 grid.
- Taking turns, each player announces where on the opponent's grid they wish to fire.
- The opponent must announce whether or not one of the ships was hit.
- The first player to sink all the opponent's ships wins.

Requirements

1. Game Setup
    a. Board size
        i. 10x10
        ii. The columns are denoted by letters (A-J)
        iii. The rows are denoted by number (1-10)
    b. Number of ships (per player)
        i. Given by user
        ii. Minimum of 1 and a maximum of 5
    c. Types of ships
        i. This will be based on the number of ships chosen.
        ii. If a total of 1 ship is chosen, then each player gets a single 1x1 ship
        iii. If a total of 2 ships is chosen, then each player gets a 1x1 and a 1x2 ship
        iv. This continues up to 5, where each player will a 1x1, 1x2, 1x3, 1x4, and a 1x5 ship
    d. Ship placement
        i. After the number of ships is chosen, players need to be able to secretly orient and place the ships on their board.
2. Playing the Game
    a. Taking turns, the users pick a space on the opponent's board to "fire" at.

   b. They must then be informed if the shot was a "hit" or a "miss".
   c. The player's view should be updated to reflect this (see Player's view below).
   d. After each shot, it is the other players turn.
3. Destroying a ship
   a. Once a ship has been hit in every space it occupies, it is sunk.
   b. For example, if the 1x3 ship occupies spaces B3, B4, and B5.
   c. Once the opponent has shot those three spaces, that ship is sunk.
4. Player's view
   a. A player should have full view of their board and where their ships are placed.
   b. Show how many times each ship has been hit
   c. A player should have a board to track all shots they've fired and whether they were misses or hits.
5. Game End
   a. Once a player has sunk all the opponent's ships, they immediately win.

Language and Platform

- Team’s choice
- Email your team’s choice to Dr. Johnson and your GTA as soon as you determine it.

Grading Rubric

- Team Score (75 points – team based)
  - This portion of the project will be graded by your GTA.
  - The project points are broken down into the following sections.
  - Demo (40 points)
    - You will demo on a device of your choice in your weekly GTA/Team meeting.
    - Specified features present
    - Stability - GTA will stress test your application; crashes, memory leaks, or other things that you also hate in bad software will be met with a penalty
    - User interface - your product should be intuitive to use; good rule to stick by: If I need a manual to use your interface, you have a bad interface.
    - Modularity - your code should be easily extensible and divided into logical components (i.e. not one class that does everything)
  - System Documentation (25 points)
    - On your GitHub repository, have a folder called "documentation" that contains all the system documentation described below.
    - Estimate of person-hours for completing the project (this should be done as soon as possible – 10 points)
      - Provide the details of how you arrived at the estimate
    - Actual accounting of the person-hours required to complete the project (10 points)
      - This needs to be a day-by-day accounting from each team member on how many hours they spent on the project, including team and GTA meetings,

        coding, testing, documenting, etc. Do not include time attending 581 lectures, working on in-class problems.
        - You WILL NOT be penalized for taking longer (or shorter) than your estimate.
        - You WILL be penalized, if you do not have a day-to-day accounting or it looks like to the GTA that you created the accounting just to meet this requirement.
    - Documentation of your code (5 points)
        - This can be in any format you choose, but it should be detailed enough that the GTA and the team who will receive your code for Project 2 can understand what you did.
        - Write it like you would like to receive it from the team whose code you get for Project 2.
        - This documentation should supplement the comments in the code.
        - As we learn about more documentation artifacts during the semester, these documentation requirements will become more specific.
- Comments (10 points)
    - The software must be adequately commented with prologue comments, comments summarizing major blocks of code, and comments on every line.
    - Adequate prologue comments include:
        - Name of program contained in the file
        - Brief description of the program
        - Inputs
        - Output
        - Other sources for the code ChatGPT, stackOverflow, etc.
        - Author's full name
        - Creation date: The date you first create the file, i.e., the date you write this comment
    - Adequate comments summarizing major blocks of code and comments on every line:
        - Provide comments that explain what each line of code is doing.
        - You may comment each line of code and/or provide a multi-line comment that explains what a group of lines does.
        - Multi-line comments should be detailed enough that it is clear what each line of code is doing.
        - Each block of code must indicate whether you authored the code or you obtained it from one of the sources listed in the prologue, or if it was a combination of both.
    - Other sources for code:
        - When you use other sources for the code (e.g., ChatGPT, stackOverflow):
        - Your comments must be significantly different from the source's comments.
        - More scrutiny will be applied to grading your comments in particular explaining the code "in your own words", not the source's comments (e.g., ChatGPT's comments).

            - Failure to identify other sources of code will not only result in a 0 on the assignment but will be considered an act of Academic Misconduct.
            - Students who violate conduct policies will be subject to severe penalties, up through and including dismissal from the School of Engineering.
        - The comments should be detailed enough that the GTA and the team who will receive your code for Project 2 can understand what you did.
        - Comment it like you would like to receive it from the team whose code you get for Project 2.

- Team Peer Evaluations (25 points – individual based)
    - This part will be based on the Team Peer Evaluation forms as described in the first lecture.
    - You may receive more than 25 points if your teammates think you did more than your fair share of the work.
    - If you turn the Team Peer Evaluation form in late or do not turn it in:
        - You will still get a Team Peer Evaluation score based on your teammate's evaluation.
        - You will lose 25 points, which could result in a negative score.
        - The $10,000 bonus you were supposed to divide between your teammates will be divided equally among your teammates.

Project 1 grading rubric:

Project 1 (75% of grade – team based)				
Measure	Points	Grading Level		
		Exceeds Expectations (90-100%)	Meets Expectations (80-89%)	Unsatisfactory (0-79%)
Demo	40	All four system requirements are met: 1) specified features present; 2) system is stable; 3) user interface is intuitive to use; and 4) code is modular.	Only three of the system requirements are met.	Only two or less of the system requirements are met.
Estimate of person-hours for completing the project	10	Details of how you arrived at the estimate are complete and easy to understand.	Details of how you arrived at the estimate are inadequate and hard to understand.	No estimate provided (0 points); or estimate provided without an explanation (60 points).
Actual accounting of the person-hours required to complete the project	10	Complete day-by-day accounting from each team member on how many hours they spent on the project, including team and GTA meetings, coding, testing, documenting, etc. not including time attending 581 lectures, working on in-class problems, etc.	Incomplete day-by-day accounting from each team member on how many hours they spent on the project, or the accounting included time not considered part of the project (e.g., time attending 581 lectures, working on in-class problems)	Otherwise
Documentation	5	Detailed enough that the GTA and the team who will receive your code for Project 2 can understand what you did.	Missing minor details	Missing significant details
Comments	10	Software is adequately commented with prologue comments, comments summarizing major blocks of code, and comments on every line.	Prologue comments are present but missing some items or some major blocks of code are not commented or there are inadequate comments on each line.	Prologue comments are missing all together or there are no comments on major blocks of code or there are very few comments on each line.