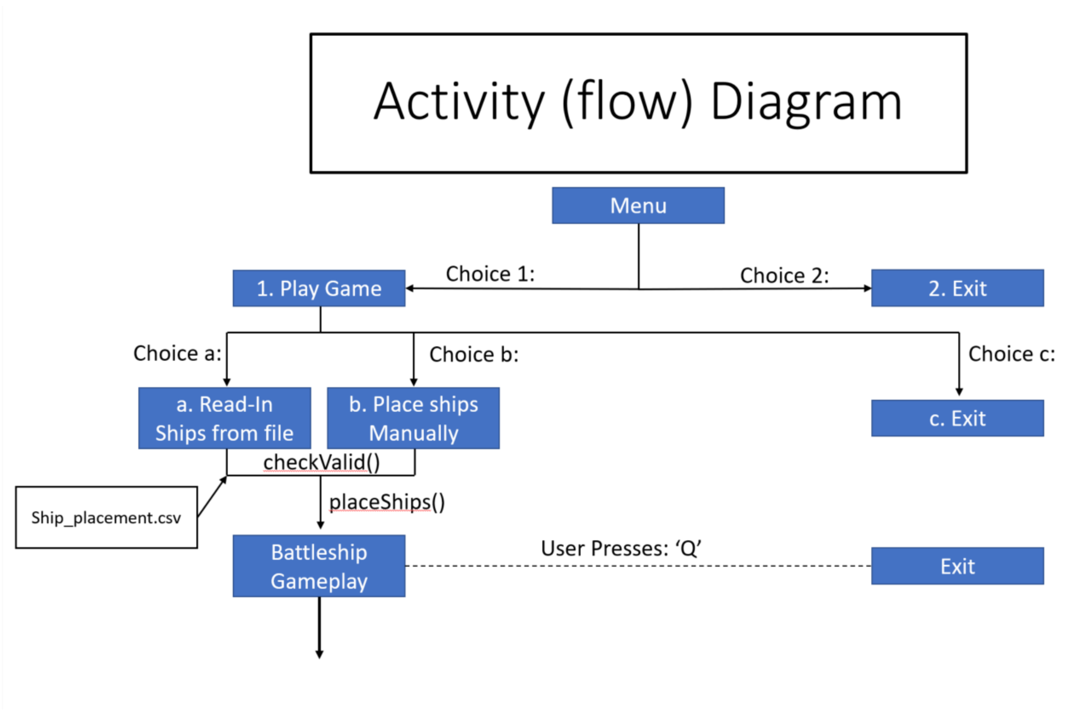
DESIGN DOCUMENT

TITLE: Battleship Program

Author: Lucas Fulmer

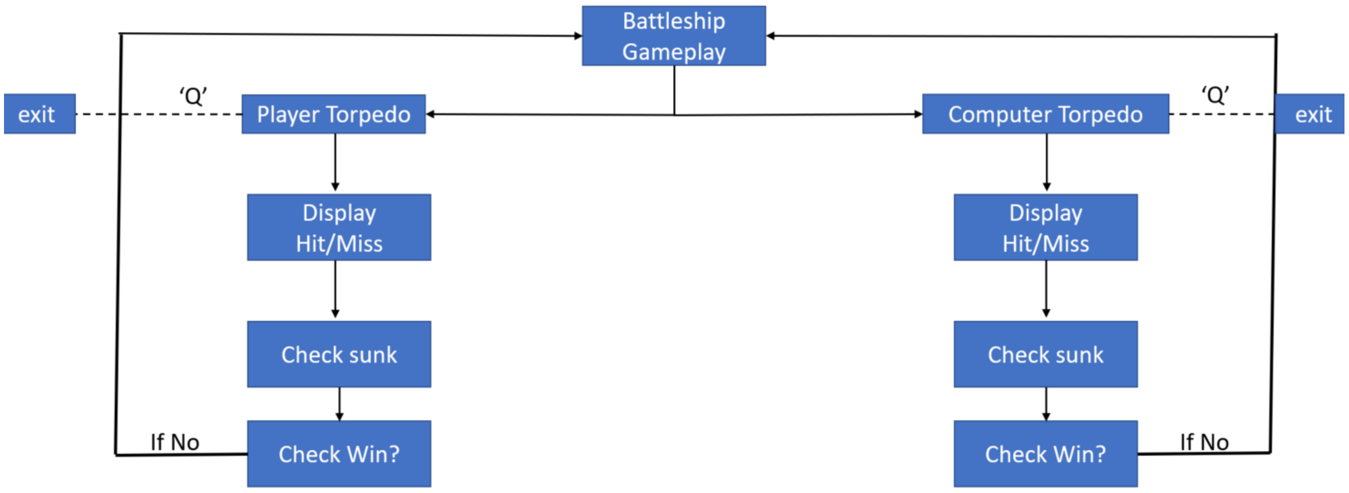
Problem Description: This program creates a "Battleship" style game. The game requires the creating of multiple 10x10 grids, where the user and the computer opponent can place his/her ships. Each player also has a second grid which shows the location of torpedoes fired as well as "hits" to enemy ships. The game terminates when one player has destroyed or "sunk" all of the opponent's ships. The program will read-in ship locations from a file for the user or allow the user to place them manually.

Overall Software Description: The program contains three classes. The first is class "Player" which has two derived classes. The second class "Grid" is used to create the "Board" and displays ships. The final class is "WaterVehicle" which creates the ships. Grid is responsible for creating up to four 10x10 grids and displaying the ship and torpedo locations. Player has multiple functions which allow the players to choose coordinates for torpedoes and to output the grids. WaterVehicle creates the ships for both players and has functions which determine ship's size, starting location, orientation, and whether the ship is sunk.

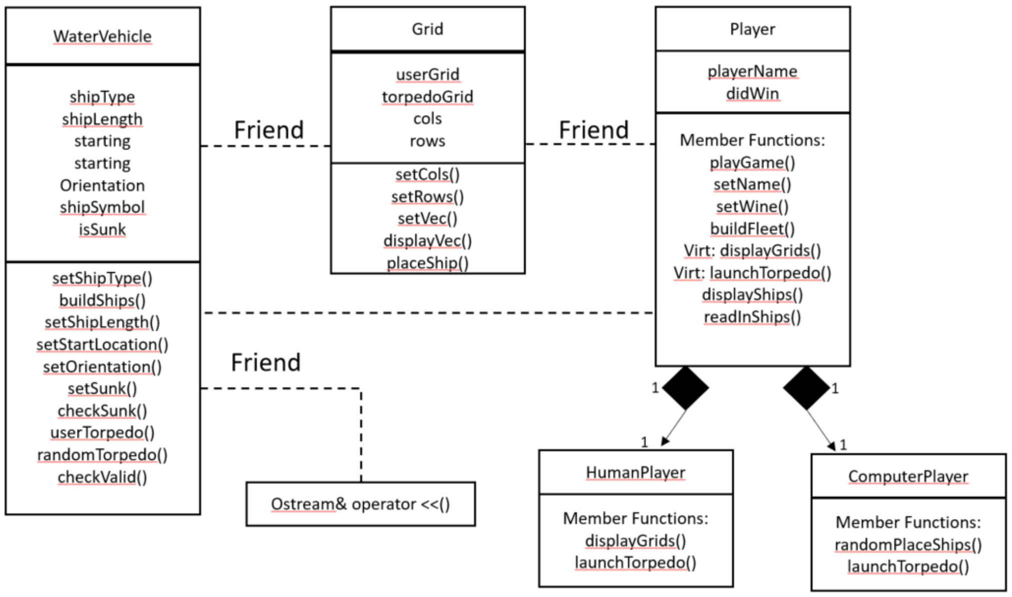


Activity Diagram continued on next page…

Activity Diagram Cont'd



Class Diagram



Input Requirements

Main Menu:

Keyboard input (requires user to choose an integer of 1. Play Game or 2. Exit)

Ship Placement:

Keyboard input (requires user to choose integer 1, 2, or 3)

1. Manual ship placement
    a. Keyboard input (user chooses upper or lowercase character ‘a’ – ‘j’)
    b. Keyboard input (user chooses integer 1 – 10)
    c. Keyboard input (user chooses upper or lower character ‘v’ or ‘h’
2. File read in (ship_placement.asv)
3. Exit program

GamePlay:

User prompted for torpedo location –

Keyboard input (user chooses upper or lowercase character ‘a’ – ‘j’)

Keyboard input (user chooses integer 1 – 10)

Optional keyboard input (user enters character ‘q’ to quit game)

Output Requirements

Main Screen

Print menu choices

Ship Placement

Print ship placement options

a. Read in ship location from file
    1. Print 10 x 10 grid showing ship locations
b. Manually place ships
    1. Print 10 x 10 grid after each ship is placed

Gameplay

Print user 10 x 10 grid with ships

Print blank 10 x 10 torpedo grid

Print hit, miss, and sunk when applicable

End of Game

Print ship statuses (name, length, starting grid, isSunk)

Problem Solution Statement

In order to develop this program, I am using three classes: Player, Grid, and WaterVehicle. When the program begins, it creates five WaterVehicles according to the Battleship game. It then allows for the user to choose whether to place ships manually or to read in the locations from a file. Regardless of the users decision, the function place_ships() will be called which passes the private members starting X, starting Y, and orientation (from WaterVehicle) into the user's grid. After all ships are placed on the user's grid, ships will be placed randomly for the computer. The program will alternate between user input torpedo shots and randomly generated computer shots, until all ships are destroyed or the user quits the game.

Classes, Inheritance, and Data Structures

This program uses three main classes: Player, Grid, and WaterVehicle. The Player class also has two child classes: HumanPlayer and ComputerPlayer. For the creation of grids, I chose to use vector data structures. I chose vectors as opposed to arrays because, in earlier versions of the game I allowed the user to specify the size of the grid he/she would use. Because the data structure will primarily be used with indexing, an array or vector would have had the same complexity. I chose to continue using the vector, out of personal choice.