Simple Battleship

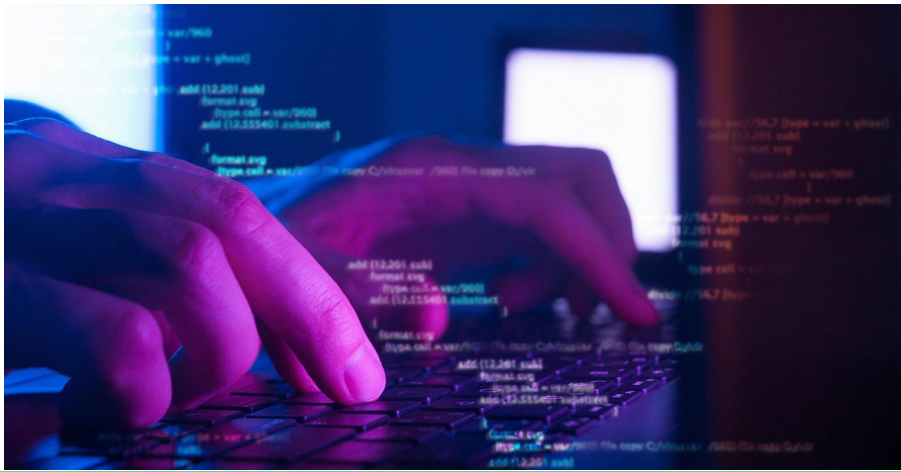
By: Simple Sam.

Overview

- What is Battleship?
  - A strategy guessing game meant for two (2) players!
  - Origins tied to French Game *L'Attaque*
  - There are 4 grids in total, 2 for each player.
  - 5 pieces for each player.
  - A player wins once all of their opponent's ships have sunk.
- Goals
  - Designing & Implement the classic Battleship game.
  - Possibly implement the game to support over 2 players.

Project Design - Technical

- Project is based around the client-server model.
  - The client-server model is an application architecture.
  - In this model there are clients & servers.
- Header files
  - network_packet.h
    - Given struct for project.
    - Altered slightly.
  - grid.h
    - Defines the two grids seen in client.
    - Has key vital functions for gameplay.



network_packet.h

```
#define OPCODE_CONNECT   0
#define OPCODE_SHOT      1
#define OPCODE_MISS      2
#define OPCODE_HIT       3
#define OPCODE_HIT_SUNK  4

#define SHIP_CARRIER     1
#define SHIP_BATTLESHIP  2
#define SHIP_DESTROYER   3
#define SHIP_SUBMARINE   4
#define SHIP_PATROL_BOAT 5

You, 3 hours ago | 2 authors (github-classroom[bot] and others)
typedef struct {
    char player_num;
    char opcode;
    char x;
    char y;
    char ship_type;
    int numOfPlayers;
    int turn;
} network_packet;

#endif
```

grid.h

```
class grid {
    public:
        // char 2D found in both subclasses.
        char theGrid_ [10][10];
};

You, 21 seconds ago | 1 author (You)
class oceanGrid : public grid {
    public:
        //vectors that hold all the coordinates of the pieces placed on the grid.
        std::vector<std::pair<int, int>> carrierPos;
        std::vector<std::pair<int, int>> battleshipPos;
        std::vector<std::pair<int, int>> destroyerPos;
        std::vector<std::pair<int, int>> submarinePos;
        std::vector<std::pair<int, int>> patrolPos;

        //constructor fills ocean grid with default value "-"
        oceanGrid(){···

        //functions print the current state of grid
        void printGrid(){···

        // when the player attempts to place a piece, function is used to validate if the player's move is legal.
        bool canUpdate(int x, int y, int shipType, char place){···

        // if the player's suggest move is legal, this function actually updates the function
        void updateGrid(int x, int y, int shipType, char place){···

        // function in charge of prompting the player to input coordinates, and calling other functions
        void configGrid(){···

        // function figures out if the ship that has been hit, is sunk
        bool shipSunk(char ship){···

        // function to figure out if the shot from the opponent has hit a ship on ocean grid
        int isHit(int x, int y, network_packet* packet){···

        // function to handle the shot given by opponent. Reports to player if the shot is a miss, hit, or hit & sunk
        network_packet handleShot(network_packet* pkt, int* SR){···
};
```

grid.h (continued)

```
class targetGrid : public grid {
    public:
        // string array to store the color of "W" in grid
        std::string targetGridColor[100];

        // constructor configure the targetGrid, along with associating a color with each "W". Colors generated selected randomly.
        targetGrid(){···

        //functions print the current state of grid
        void printGrid(){···

        // after shooting a shot, this function is called to report the status of the shot (miss, hit, hit & sunk)
        void handleResponse(network_packet* pkt, int* SS){···
};

#endif
```

Challenges/Triumph

- Key Challenges:
  - Turn Management.
  - Tracking number of players.
- Solutions:
  - Modification of the network_packet.
  - Slight modification of server.

Demo

https://youtu.be/K1HaNBaWYH0

Final Thoughts