

CSC 470 Report for Project: 03

Full Name: Sam Blamo

Section: 01

1. Introduction

a. Goal

The primary objective of this project was to develop a network-based, multiplayer Battleship game using C++ programming. The game aimed to replicate the classic Battleship experience, where players place ships on a grid and take turns guessing their opponent's ship locations. The challenge was to create a functional and interactive game environment over a network, using client-server architecture.

b. Problems

The primary challenge of this project was to manage complex game states and synchronize them across multiple clients in a networked environment. Implementing a robust and efficient communication protocol between the server and clients was essential. Additionally, handling concurrency and ensuring consistent game state amidst simultaneous player actions added to the complexity. The difficulty of the task lay in the intricacies of socket

programming, multithreading, and the logic required to manage a multiplayer game environment.

2. Approach

The approach taken was to split the game into two main components: the server and the client. The server was responsible for maintaining the game state, managing client connections, and relaying messages between clients. The client was designed to provide a user interface for the player to interact with the game, send commands to the server, and receive updates.

This design was expected to work well in a local network environment where latency is minimal. The client-server model was chosen for its scalability and ease of managing game states centrally.

3. Methodology

a. Implementation Pieces

The project was divided into several key components:

1. Grid Management (grid.h): Implementing the game board for each player.
2. Network Communication (network_packet.h, server.cpp, test_client.cpp): Handling the exchange of data between the server and clients.

b. Grid Management

Implementations: The grid was implemented as a class with methods to initialize, update, and display the grid.

Choice: A class-based implementation was chosen for its encapsulation and ease of managing the game state.

c. Network Communication

Server (server.cpp): Implemented using socket programming.

- Choice: Single-threaded server with select() for handling multiple clients

Client (test_client.cpp): Implemented with multithreading to handle user input and network communication simultaneously.

- Choice: Multithreading was necessary to keep the user interface responsive while waiting for network messages

d. Difficulties and Choices

Turns Feature: Implementing turns in a networked environment was challenging. The solution involved tracking player turns on the server and synchronizing this information with clients.

Tracking Number of Players: To manage the game flow based on the number of connected players, the server needed to dynamically track active connections and update clients accordingly.

4. Results

a. Testing Objectives

Turn Management: Ensuring that turns were correctly managed and synchronized across clients.

Player Connection Management: Verifying that the server correctly handled varying numbers of players and their states.

b. Experiments

Varying Number of Players: Tested with different numbers of clients connected to the server to observe behavior and stability.

Turn Sequence Testing: Tested the turn-taking mechanism under normal and edge-case scenarios.

c. Outcomes

The results indicated that the game managed turns effectively and handled multiple players with stability. However, issues arose in edge cases, such as abrupt client disconnections.

5. Discussion

a. Opinions

The approach was largely successful in creating a functional multiplayer Battleship game. However, there is room for improvement in handling edge cases and enhancing the user interface.

b. Areas for improvement

Improved Turn Management: Implementing a more robust system to handle unexpected scenarios like client disconnections.

User Interface: Enhancing the client interface for a more engaging user experience.

c. Possible Features

Implementing additional game features like chat or advanced game statistics.

d. Learning Experience

Although I was familiar with socket programming, this was the first time using it to make a multiplayer game on a network. With this experience, I've learned the complex nature of real-time gaming.

6. Conclusion

This Battleship game project demonstrates a successful implementation of a classic game in a networked environment using C++. The challenges faced, particularly in turn management and tracking player numbers, were significant learning experiences. The project underscores the importance of robust design and testing in networked applications. Future work can expand on this foundation to create a more feature-rich and scalable multiplayer game.

7. Source Code

network_packet.h

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code, Dec 18 03:32
- File Explorer:** Shows a tree view with files: grid.h, network_packet.h (selected), and network_packet.h ...
- Search Bar:** enter x
- Code Editor:** Displays the content of network_packet.h. The code defines a struct for Network Packet Specification, including fields for player_num (0..N), opcode (0..10), and ship_type (1..5). It also defines OPCODE_CONNECT through OPCODE_HIT_SUNK and SHIP_CARRIER through SHIP_PATROL_BOAT.
- Bottom Status Bar:** Shows file paths, line numbers (L13, Col 50), spaces, encoding (UTF-8), C/C++, Go Live, spell checker status, Linux, Tabnine starter, and a prettier button.

```
#ifndef NETWORK_PACKET
#define NETWORK_PACKET

Network Packet Specification
player_num
    0..N - player number
opcode
    0 - connected to server as player_num
    1 - shot from player_num at (x,y)
    2 - hit from player_num ship_type at (x,y)
    3 - hit from player_num ship_type at (x,y)
    4 - hit from player_num and sunk ship_type at (x,y)
    5 - coordinate
    6 - y coordinate
    7 - x coordinate
    8 - y coordinate
    9 - x coordinate
    10 - y coordinate
ship_type
    1 - carrier (5 holes)
    2 - battleship (4 holes)
    3 - destroyer (3 holes)
    4 - submarine (3 holes)
    5 - patrol boat (2 holes)

#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
```

grid.h

```
grid.h  M  h network_packet.h
grid.h  > oceanGrid > configGrid()
1 #ifndef GRID
2 #define GRID
3
4 #include "network_packet.h"
5 #include <cstdlib>
6 #include <string>
7 #include <iostream>
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <string.h>
11 #include <vector>
12 #include <unistd.h>
13 #include <utility>
14
15 You, yesterday | author (You)
16 class grid {
17 public:
18     char theGrid_[10][10];
19 };
20
21 You, 9 minutes ago | author (You)
22 class oceanGrid : public grid {
23 public:
24     std::vector<std::pair<int, int>> carrierPos;
25     std::vector<std::pair<int, int>> battleshipPos;
26     std::vector<std::pair<int, int>> destroyerPos;
27     std::vector<std::pair<int, int>> submarinePos;
28     std::vector<std::pair<int, int>> patrolPos;
29
30     oceanGrid(){
31         for(int i = 0; i < 10; i++){
32             for(int j = 0; j < 10; j++){
33                 this->theGrid_[i][j] = '-';
34             }
35         }
36     }
37
38     void printGrid(){
39         for(int i = 0; i < 10; i++){
40             for(int j = 0; j < 10; j++){
41                 if(this->theGrid_[i][j] == 'X'){
42                     printf("\033[0;31m%c ", this->theGrid_[i][j]);
43                 } else if(this->theGrid_[i][j] != '-' && this->theGrid_[i][j] != 'X' ){
44                     printf("\033[0;32m%c ", this->theGrid_[i][j]);
45                 } else {
46                     printf("%c ", this->theGrid_[i][j]);
47                 }
48             }
49         }
50     }
51 }
```

```
grid.h - battleship-Simon-Blamo - Visual Studio Code

File Edit Selection View Go Run Terminal Help
grid.h > network_packet.h
grid.h > configGrid()
grid.h > oceanGrid > configGrid()

50     printf("\n");
51 }
52 printf("\n\n");
53 }

54 bool canUpdate(int x, int y, int shipType, char place){
55     if( x < 0 || x >= 10 ) {
56         printf("Invalid X coordinate!\n\n");
57         return false;
58     }
59     if( y < 0 || y >= 10 ) {
60         printf("Invalid Y coordinate!\n\n");
61         return false;
62     }
63     if ((place == 'H' || place == 'h') || (place == 'V' || place == 'v'))(
64         printf("Invalid placement input! Please try again!\n\n");
65         return false;
66     }

67     if(shipType == 1){
68         if(place == 'H' || place == 'h'){
69             if((x) + 5 > 9 ){
70                 printf("Ship placement would be out of bounds! Please try again!\n\n");
71                 return false;
72             }
73         } else if(place == 'V' || place == 'v'){
74             if((y) + 5 > 9 ){
75                 printf("Ship placement would be out of bounds! Please try again!\n\n");
76                 return false;
77             }
78         }
79     }
80     } else if(shipType == 2){
81         if(place == 'H' || place == 'h'){
82             if((x) + 4 > 9 ){
83                 printf("Ship placement would be out of bounds! Please try again!\n\n");
84                 return false;
85             }
86             for(int i = x; i < (x) + 4; i++){
87
88                 if(this->theGrid[y][i] != '-'){
89                     printf("A ship is present at this location! Please try again!\n\n");
90                     return false;
91                 }
92             }
93             printf("\n");
94         } else if(place == 'V' || place == 'v'){
95             if((y) + 4 > 9 ){
96                 printf("Ship placement would be out of bounds! Please try again!\n\n");
97             }
98         }
99     }
100 }
```

Activities Visual Studio Code

grid.h > oceanGrid > configGrid()

```
145     if(this->theGrid_[y][i] != '-'){
146         printf("A ship is present at this location! Please try again!\n\n");
147         return false;
148     }
149     else if(place == 'V' || place == 'V'){
150         if(y + 3 > 9){
151             printf("Ship placement would be out of bounds! Please try again!\n\n");
152             return false;
153         }
154         for(int i = y; i < (y + 3); i++){
155             if(this->theGrid_[i][x] != '-'){
156                 printf("A ship is present at this location! Please try again!\n\n");
157                 return false;
158             }
159         }
160     }
161     // for(int i = x; i < x + 3; i++){
162     //     if(theGrid_[i][y] != '-') return false;
163     // }
164     else {
165         if(place == 'W' || place == 'H'){
166             if(x + 2 > 9){
167                 printf("Ship placement would be out of bounds! Please try again!\n\n");
168                 return false;
169             }
170             for(int i = x; i < (x + 2); i++){
171                 if(this->theGrid_[y][i] != '-'){
172                     printf("A ship is present at this location! Please try again!\n\n");
173                     return false;
174                 }
175             }
176         }
177         else if(place == 'V' || place == 'V'){
178             if(y + 2 > 9){
179                 printf("Ship placement would be out of bounds! Please try again!\n\n");
180                 return false;
181             }
182             for(int i = y; i < (y + 2); i++){
183                 if(this->theGrid_[i][x] != '-'){
184                     printf("A ship is present at this location! Please try again!\n\n");
185                     return false;
186                 }
187             }
188         }
189     }
190     return true;
191 }
```

You, yesterday Ln 353, Col 27 Spaces: 4 UTF-8 LF ↴ C++ Go Live Quokka Spell Linux Tabline starter Prettier

Activities Visual Studio Code

grid.h > oceanGrid > configGrid()

```
193 void updateGrid(int x, int y, int shipType, char place){
194     std::pair<int, int> temp;
195     if(shipType == 1){
196
197         if(place == 'W' || place == 'H'){
198             for(int i = y; i < (y + 5); i++){
199                 this->theGrid_[y][i] = 'C';
200                 temp.first = i;
201                 temp.second = y;
202                 this->carrierPos.push_back(temp);
203             }
204         }
205         else {
206             for(int i = y; i < (y + 5); i++){
207                 this->theGrid_[i][x] = 'C';
208                 temp.first = x;
209                 temp.second = i;
210                 this->carrierPos.push_back(temp);
211             }
212         }
213     }
214     else if(shipType == 2){
215         if(place == 'W' || place == 'H'){
216             for(int i = x; i < (x + 4); i++){
217                 this->theGrid_[y][i] = 'B';
218                 temp.first = i;
219                 temp.second = y;
220                 this->battleshipPos.push_back(temp);
221             }
222         }
223         else {
224             for(int i = y; i < (y + 4); i++){
225                 this->theGrid_[i][x] = 'B';
226                 temp.first = x;
227                 temp.second = i;
228                 this->battleshipPos.push_back(temp);
229             }
230         }
231     }
232     else if(shipType == 3){
233         if(place == 'W' || place == 'H'){
234             for(int i = x; i < (x + 3); i++){
235                 this->theGrid_[y][i] = 'D';
236                 temp.first = i;
237                 temp.second = y;
238                 this->destroyerPos.push_back(temp);
239             }
240         }
241         else {
242             for(int i = y; i < (y + 3); i++){
243                 this->theGrid_[i][x] = 'D';
244                 temp.first = x;
245                 temp.second = i;
246                 this->destroyerPos.push_back(temp);
247             }
248         }
249     }
250 }
```

You, yesterday Ln 353, Col 27 Spaces: 4 UTF-8 LF ↴ C++ Go Live Quokka Spell Linux Tabline starter Prettier

Activities Visual Studio Code Dec 18 03:35 13%

File Edit Selection View Go Run Terminal Help

grid.h network_packet.h

```
240     this->theGrid_[i][x] = 'D';
241     temp.first = x;
242     temp.second = i;
243     this->destroyerPos.push_back(temp);
244 }
245 } else if(shipType == 4){
246     if(place == 'H' || place == 'h'){
247         for(int i = x; i < (x + 3); i++){
248             this->theGrid_[y][i] = 'S';
249             temp.first = i;
250             temp.second = y;
251             this->submarinePos.push_back(temp);
252         }
253     } else {
254         for(int i = y; i < (y + 3); i++){
255             this->theGrid_[i][x] = 'S';
256             temp.first = x;
257             temp.second = i;
258             this->submarinePos.push_back(temp);
259         }
260     }
261 } else {
262     if(place == 'H' || place == 'h'){
263         for(int i = x; i < (x + 2); i++){
264             this->theGrid_[y][i] = 'P';
265             temp.first = i;
266             temp.second = y;
267             patrolPos.push_back(temp);
268         }
269     } else {
270         for(int i = y; i < (y + 2); i++){
271             this->theGrid_[i][x] = 'P';
272             temp.first = x;
273             temp.second = i;
274             patrolPos.push_back(temp);
275         }
276     }
277 }
278 }
279 }
280 }
281 void configGrid(){
282     int posX;
283     int posY;
284     char buffer[100];
285     char placement;
286     bool wasValid;
287 }
```

You, yesterday Ln 353, Col 27 Spaces: 4 UTF-8 LF ⓘ C++ Go Live Quokka Spell Linux Tabline starter Prettier

Activities Visual Studio Code Dec 18 03:35 13%

File Edit Selection View Go Run Terminal Help

grid.h network_packet.h

```
288     printf("\n\nWelcome to Battleship! Let's place your pieces to start the game!\n");
289     printf("Place your Carrier:\n\n");
290     fgets(buffer, 100, stdin);
291     sscanf(buffer, "%d %d %c", &posX, &posY, &placement);
292     while(true){
293         printf("Enter the ship's X, Y coordinates, along with it's placement! Use H to place the piece horizontally, and user V to place the ship vertically! (e. g. 1 2 H: ");
294         fgets(buffer, 100, stdin);
295         sscanf(buffer, "%d %d %c", &posX, &posY, &placement);
296         posX--;
297         posY--;
298         wasValid = canUpdate(posX, posY, 1, placement);
299         if(wasValid){
300             this->updatedGrid(posX, posY, 1, placement);
301             this->printGrid();
302             break;
303         }
304     }
305     memset(buffer, '\0', sizeof(buffer));
306     printf("Place your Battleship:\n\n");
307     while(true){
308         printf("Enter the ship's X, Y coordinates, along with it's placement: ");
309         fgets(buffer, 100, stdin);
310         sscanf(buffer, "%d %d %c", &posX, &posY, &placement);
311         posX--;
312         posY--;
313         wasValid = canUpdate(posX, posY, 2, placement);
314         if(wasValid){
315             this->updatedGrid(posX, posY, 2, placement);
316             this->printGrid();
317             break;
318         }
319     }
320     memset(buffer, '\0', sizeof(buffer));
321     printf("Place your Destroyer:\n\n");
322     while(true){
323         printf("Enter the ship's X, Y coordinates, along with it's placement: ");
324         fgets(buffer, 100, stdin);
325         sscanf(buffer, "%d %d %c", &posX, &posY, &placement);
326         posX--;
327         posY--;
328         wasValid = canUpdate(posX, posY, 3, placement);
329         if(wasValid){
330             this->updatedGrid(posX, posY, 3, placement);
331             this->printGrid();
332             break;
333         }
334     }
335 }
```

You, yesterday Ln 353, Col 27 Spaces: 4 UTF-8 LF ⓘ C++ Go Live Quokka Spell Linux Tabline starter Prettier

```
Activities > Visual Studio Code Dec 18 03:35 ▾
grid.h - battleship-Simon-Blamo - Visual Studio Code

File Edit Selection View Go Run Terminal Help

h grid.h M h network_packet.h
h grid.h > oceanGrid > configGrid()
333     posX--;
334     posY--;
335     wasValid = canUpdate(posX, posY, 3, placement);
336     if(wasValid){
337         this->updateGrid(posX, posY, 3, placement);
338         this->printGrid();
339         break;
340     }
341 }
342 memset(buffer, '\0', sizeof(buffer));
343 printf("Place your Submarine!\n\n");
344
345 while(true){
346     printf("Enter the ship's X, Y coordinates, along with it's placement!: ");
347     fgets(buffer, 100, stdin);
348     sscanf(buffer, "%d %d %c", &posX, &posY, &placement);
349     posX--;
350     posY--;
351     wasValid = canUpdate(posX, posY, 4, placement);
352     if(wasValid){
353         this->updateGrid(posX, posY, 4, placement);
354         this->printGrid();
355         break;
356     }
357 }
358
359 memset(buffer, '\0', sizeof(buffer));
360 printf("Place your Patrol Boat!\n\n");
361
362 while(true){
363     printf("Enter the ship's X, Y coordinates, along with it's placement!: ");
364     fgets(buffer, 100, stdin);
365     sscanf(buffer, "%d %d %c", &posX, &posY, &placement);
366     posX--;
367     posY--;
368     wasValid = canUpdate(posX, posY, 5, placement);
369     if(wasValid){
370         this->updateGrid(posX, posY, 5, placement);
371         system("clear");
372         printf("OCEAN GRID\n");
373         this->printGrid();
374         break;
375     }
376 }
377
378 printf("\n\n");
379 // printf("C: %d, B: %d, D: %d, S: %d, P: %d\n", (int)this->carrierPos.size(), (int)this->battleshipPos.size(), (int)this->destroyerPos.size(), (int)this->submarinePos.size(), (int)
380 this->patrolPos.size());
381
382 main* D0 D0 D0 W0 Live Share ▾
```

Activities Visual Studio Code

grid.h battleship-Simon-Blamo-Visual Studio Code

```
File Edit Selection View Go Run Terminal Help
```

grid.h > oceanGrid > configGrid()

```
428     return 2;
429 }
430 else if(this->theGrid_[y][x] == 'B'){
431     for(int i = 0; i < (int)this->battleshipPos.size(); i++){
432         if(this->battleshipPos[i].first == x && this->battleshipPos[i].second == y){
433             this->battleshipPos.erase(this->battleshipPos.begin() + i);
434             break;
435         }
436     }
437
438     if(shipSunk('B')){
439         theGrid_[y][x] = 'X';
440         packet->ship_type = 2;
441         return 2;
442     }
443     else if(this->theGrid_[y][x] == 'D'){
444         for(int i = 0; i < (int)this->destroyerPos.size(); i++){
445             if(this->destroyerPos[i].first == x && this->destroyerPos[i].second == y){
446                 destroyerPos.erase(this->destroyerPos.begin() + i);
447                 break;
448             }
449         }
450
451         if(shipSunk('D')){
452             this->theGrid_[y][x] = 'X';
453             packet->ship_type = 3;
454             return 2;
455         }
456         else if(this->theGrid_[y][x] == 'S'){
457             for(int i = 0; i < (int)this->submarinePos.size(); i++){
458                 if(this->submarinePos[i].first == x && this->submarinePos[i].second == y){
459                     submarinePos.erase(this->submarinePos.begin() + i);
460                     break;
461                 }
462             }
463
464             if(shipSunk('S')){
465                 this->theGrid_[y][x] = 'X';
466                 packet->ship_type = 4;
467                 return 2;
468             }
469         }
470         else if(this->theGrid_[y][x] == 'P'){
471             for(int i = 0; i < (int)this->patrolPos.size(); i++){
472                 if(this->patrolPos[i].first == x && this->patrolPos[i].second == y){
473                     patrolPos.erase(this->patrolPos.begin() + i);
474                     break;
475                 }
476             }
477         }
478     }
479     else if(shipSunk('P')){
480         this->theGrid_[y][x] = 'X';
481         return 1;
482     }
483 }
484 }
```

You, yesterday Ln 353, Col 27 Spaces: 4 UTF-8 LF ↴ C++ Go Live Quokka Spell Linux Tabline starter Prettier

Activities Visual Studio Code

grid.h battleship-Simon-Blamo-Visual Studio Code

```
File Edit Selection View Go Run Terminal Help
```

grid.h > oceanGrid > configGrid()

```
476     if(shipSunk('P')){
477         this->theGrid_[y][x] = 'X';
478         packet->ship_type = 5;
479         return 2;
480     }
481     this->theGrid_[y][x] = 'X';
482     return 1;
483 }
484 }
```

network_packet handleShot(network_packet* pkt, int* SR){

```
485     network_packet* returnPkt = new network_packet();
486     int x = (pkt->x)-1;
487     int y = (pkt->y)-1;
488     if(pkt->player_num == 0){
489         returnPkt->x = pkt->x;
490         returnPkt->y = pkt->y;
491         returnPkt->ship_type = 0;
492         // printf(" Packet received: %hd %hd %hd %hd %hd\n", pkt->opcode, pkt->player_num, pkt->ship_type, pkt->x, pkt->y);
493         // printf("X, Y: %d %d\n", x, y);
494         printf("Ship received:\n");
495         printf("Ship has...\n");
496         fflush(stdout);
497         sleep(1);
498         printf(".");
499         fflush(stdout);
500         sleep(1);
501         printf(".");
502         fflush(stdout);
503         sleep(1);
504         printf(".");
505         fflush(stdout);
506         sleep(1);
507         printf(".");
508         fflush(stdout);
509         sleep(2);
510         int hit = isHit(x, y, pkt);
511         if(hit == 0){
512             *SR -=;
513             printf("\x1b[33;0;32m MISSED!\n");
514             returnPkt->opcode = 2;
515         } else if(hit == 1){
516             *SR -=;
517             printf("\x1b[33;0;31m HIT!\n");
518             returnPkt->opcode = 3;
519         } else if(hit == 2){
520             *SR -=;
521             printf("\x1b[33;0;91m HIT! SHIP SUNK!\n");
522             returnPkt->opcode = 4;
523         }
524         printf("\x1b[33;0m");
525     }
526 }
```

You, yesterday Ln 353, Col 27 Spaces: 4 UTF-8 LF ↴ C++ Go Live Quokka Spell Linux Tabline starter Prettier

server.cpp

Activities Visual Studio Code Dec 18 03:46

File Edit Selection View Go Run Terminal Help

grid.h M C++ server.cpp

server.cpp > ...

```
1 /*****  
2 ======  
3 =====> DO NOT MODIFY THIS FILE! <=====  
4 ======  
5  
6 BattleShip server application  
7  
8 This code creates a server socket and listens for incoming connections.  
9 When a client connects, the server adds the new socket to a list of  
10 connected clients. The server then enters a loop where it waits for  
11 activity on any of the connected sockets using select(). When a message  
12 is received from a client, the server sends that message to all  
13 connected clients except the sender.  
14 | github-classroom[bot], 2 weeks ago • Initial commit  
15 | How to compile:  
16 | g++ server.cpp -o server  
17 */  
18  
19 #include <stdio.h>  
20 #include <stdlib.h>  
21 #include <string.h>  
22 #include <sys/socket.h>  
23 #include <sys/types.h>  
24 #include <unistd.h>  
25 #include <errno.h>  
26 #include "network_packet.h"  
27  
28 #define MAX_CLIENTS 30  
29 #define BACKLOG 10  
30 #define BUFFER_SIZE 100  
31 #define SERVER_PORT 8888  
32  
33 int main(int argc, char *argv[]) {  
34     int server_socket, addrlen, new_socket, client_socket[MAX_CLIENTS], activity, i, j, _valread, sd;  
35     int max_sd;  
36     struct sockaddr_in address;  
37     char buffer[BUFFER_SIZE];  
38     fd_set readfds;  
39  
40     // initialize client array  
41     for (i = 0; i < MAX_CLIENTS; i++) {  
42         client_socket[i] = 0;  
43     }  
44  
45     // create server socket  
46     if ((server_socket = socket(AF_INET, SOCK_STREAM, 0)) == 0) {  
47  
48         perror("socket failed");  
49         exit(EXIT_FAILURE);  
50     }  
51  
52     // bind server socket  
53     address.sin_family = AF_INET;  
54     address.sin_addr.s_addr = INADDR_ANY;  
55     address.sin_port = htons(SERVER_PORT);  
56     addrlen = sizeof(address);  
57     if (bind(server_socket, (struct sockaddr *)&address, addrlen) < 0) {  
58         perror("bind failed");  
59         exit(EXIT_FAILURE);  
60     }  
61  
62     // listen on server socket  
63     if (listen(server_socket, BACKLOG) < 0) {  
64         perror("listen failed");  
65         exit(EXIT_FAILURE);  
66     }  
67     printf("Server listening: socket fd %d, ip %s, port %d\n",  
68           server_socket, inet_ntoa(address.sin_addr), ntohs(address.sin_port));  
69  
70     // loop on select  
71     while (1) {  
72         // clear the file descriptor set  
73         FD_ZERO(&readfds);  
74  
75         // add server_socket to the file descriptor set  
76         FD_SET(server_socket, &readfds);  
77         max_sd = server_socket;  
78  
79         // add each client_socket to the file descriptor set  
80         for (i = 0; i < MAX_CLIENTS; i++) {  
81             sd = client_socket[i];  
82             if (sd > 0) {  
83                 FD_SET(sd, &readfds);  
84             }  
85             if (sd > max_sd) {  
86                 max_sd = sd;  
87             }  
88         }  
89  
90         // wait on select()  
91         activity = select(max_sd + 1, &readfds, NULL, NULL, NULL);  
92         if ((activity < 0) && (errno != EINTR)) {  
93             printf("select error");  
94         }  
95     }
```

github-classroom[bot], 2 weeks ago Ln 14, Col 1 Spaces: 4 UTF-8 LF ⚡ C++ Go Live ⚡ Quokka ⚡ Spell Linux Tabnine starter See Tabnine Insights ⚡ Prettier

Activities Visual Studio Code Dec 18 03:46

File Edit Selection View Go Run Terminal Help

grid.h M C++ server.cpp

server.cpp > ...

```
48     perror("socket failed");  
49     exit(EXIT_FAILURE);  
50 }  
51  
52 // bind server socket  
53 address.sin_family = AF_INET;  
54 address.sin_addr.s_addr = INADDR_ANY;  
55 address.sin_port = htons(SERVER_PORT);  
56 addrlen = sizeof(address);  
57 if (bind(server_socket, (struct sockaddr *)&address, addrlen) < 0) {  
58     perror("bind failed");  
59     exit(EXIT_FAILURE);  
60 }  
61  
62 // listen on server socket  
63 if (listen(server_socket, BACKLOG) < 0) {  
64     perror("listen failed");  
65     exit(EXIT_FAILURE);  
66 }  
67 printf("Server listening: socket fd %d, ip %s, port %d\n",  
68       server_socket, inet_ntoa(address.sin_addr), ntohs(address.sin_port));  
69  
70 // loop on select  
71 while (1) {  
72     // clear the file descriptor set  
73     FD_ZERO(&readfds);  
74  
75     // add server_socket to the file descriptor set  
76     FD_SET(server_socket, &readfds);  
77     max_sd = server_socket;  
78  
79     // add each client_socket to the file descriptor set  
80     for (i = 0; i < MAX_CLIENTS; i++) {  
81         sd = client_socket[i];  
82         if (sd > 0) {  
83             FD_SET(sd, &readfds);  
84         }  
85         if (sd > max_sd) {  
86             max_sd = sd;  
87         }  
88     }  
89  
90     // wait on select()  
91     activity = select(max_sd + 1, &readfds, NULL, NULL, NULL);  
92     if ((activity < 0) && (errno != EINTR)) {  
93         printf("select error");  
94     }  
95 }
```

github-classroom[bot], 2 weeks ago Ln 14, Col 1 Spaces: 4 UTF-8 LF ⚡ C++ Go Live ⚡ Quokka ⚡ Spell Linux Tabnine starter See Tabnine Insights ⚡ Prettier

```
96     // new connection?
97     if (FD_ISSET(server_socket, &readfds)) {
98         // accept connection
99         if ((new_socket = accept(server_socket, (struct sockaddr *)&address, (socklen_t *)&addrlen)) < 0) {
100             perror("accept failed");
101             exit(EXIT_FAILURE);
102         }
103
104         // add new connection to first empty client_socket element
105         for (i = 0; i < MAX_CLIENTS; i++) {
106             if (client_socket[i] == 0) {
107                 client_socket[i] = new_socket;
108                 break;
109             }
110         }
111         printf("Client connect: socket fd %d, ip %s, port %d, client_socket[%d]\n",
112               new_socket, inet_ntoa(address.sin_addr), ntohs(address.sin_port), i);
113
114         // upon connection, send client its player_num
115         network_packet player;
116         int count = 0;
117         while(client_socket[count] != 0){
118             count++;
119         }
120         player.numOfPlayers = count;
121         player.opcode = OPCODE_CONNECT;
122         player.player_index = i; // same index as player_num
123         player.x = player.y = player.ship.type = 0;
124         send(client_socket[i], &player, sizeof(network_packet), 0);
125         for (int qa = 0; qa < MAX_CLIENTS; qa++) {
126             if(qa != i){
127                 send(client_socket[qa], &player, sizeof(network_packet), 0);
128             }
129         }
130     }
131
132     // check each client in the file descriptor set
133     for (i = 0; i < MAX_CLIENTS; i++) {
134         sd = client_socket[i];
135
136         // is client ready for reading?
137         if (FD_ISSET(sd, &readfds)) {
138             // handle disconnection
139             if ((valread = read(sd, buffer, BUFFER_SIZE)) == 0) {
140                 printf("Client disconnect: socket fd %d, client_socket[%d]\n",
141                       client_socket[i], i);
142                 close(sd);
143                 client_socket[i] = 0;
144             }
145         }
146     }
147 }
```

The screenshot shows a Visual Studio Code interface with a dark theme. The top bar displays 'Activities' and 'Visual Studio Code'. The status bar shows the date 'Dec 18 03:46' and battery level '9%'. A tooltip 'Screenshot captured' with the message 'You can paste the image from the clipboard.' is visible. The left sidebar has icons for file operations like Open, Save, Find, and Replace. The main editor window contains a C++ file named 'server.cpp' with code related to network sockets. The code includes a loop that reads data from a socket, checks its length, and then sends it to all other clients except the sender. The code uses standard C++ libraries like `<iostream>`, `<vector>`, and `<sys/types.h>`. The bottom status bar shows file paths like 'main.cpp', 'grid.h', and 'grid.cpp', along with various status indicators such as '0 main.cpp', '0 grid.h', '0 grid.cpp', 'Live Share', 'github-classroom[bot] 2 weeks ago', 'Ln 14, Col 1', 'Spaces: 4', 'UTF-8', 'C++', 'Go Live', 'Quickfix', 'Spell', 'Linux', 'Tabnine starter', and 'See Tabnine Insights'.

```
143     }
144     // data was read
145     else {
146         // check for valid packet length
147         if (valread != sizeof(network_packet)) {
148             printf("WARNING: packet size %d from client_socket[%i] ignored\n", valread, i);
149         }
150         else {
151             // echo received packet to all clients except the sender
152             for (j = 0; j < MAX_CLIENTS; j++) {
153                 if (client_socket[j] != 0 && client_socket[j] != sd) {
154                     send(client_socket[j], buffer, valread, 0);
155                 }
156             }
157         }
158     }
159 }
160
161 return 0;
162
163
164 }
```

client.cpp

Activities Visual Studio Code Dec 18 03:47 10%

File Edit Selection View Go Run Terminal Help

grid.h M C++ client.cpp

client.cpp > tx(void)

You, 2 hours ago | 2 authors (You and others)

```
1 // Battleship test client application
2
3 This code creates a client socket and connects to the server using the
4 IP address and port number of the server. The client then enters a loop
5 where it waits for user input and sends any messages to the server.
6
7 The client also waits for incoming messages from the server and prints
8 them to the console.
9
10 How to compile:
11     g++ test_client.cpp -o test_client -lpthread
12 */
13
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include <string.h>
17 #include <pthread.h>
18 #include <sys/socket.h>
19 #include <arpa/inet.h>
20 #include <vector>
21 #include <unistd.h>
22 #include "grid.h"
23
24 #define LOCALHOST "127.0.0.1"
25 #define SERVER_PORT 8888
26 #define BUFFER_SIZE 100
27
28
29 pthread_mutex_t socket_mutex = PTHREAD_MUTEX_INITIALIZER;
30 int client_socket = -1;
31 bool connected = false;
32 int shipsSunk = 0;
33 int shipsRemaining = 5;
34 oceanGrid* theOceanGrid;
35 targetGrid* theTargetGrid;
36 int totalNumOfPlayers = 0;
37 int turn = 0;
38
39
40
41
42 // network send thread
43 void *tx(void *arg) {
44     pthread_detach(pthread_self());
45     int ret;
46     char buffer[BUFFER_SIZE];
47 }
```

Help us improve our support for C++ Take Short Survey Remind Me Later

github-classroom[bot], 2 weeks ago Ln 85, Col 2 Spaces: 4 UTF-8 LF ⌂ C++ Go Live ⌂ Quokka ⌂ Spell Linux ⌂ tabnine starter ⌂ See Tabnine Insights ⌂ prettier ⌂

Activities Visual Studio Code Dec 18 03:48 11%

File Edit Selection View Go Run Terminal Help

grid.h M C++ client.cpp

client.cpp > tx(void)

```
48
49     network_packet pkt = {};
50
51     // delay to allow recv of connect opcode
52     sleep(1);
53
54     // loop forever
55
56     pkt.player_num = player_number;
57     pkt.opcode = 0;
58     send(client_socket, &pkt, sizeof(network_packet));
59     sleep(2);
60     while (1) {
61         if(turn == player_number){
62             printf("Enter X, Y:\n");
63             fflush(stdout);
64             fgets(buffer, BUFFER_SIZE, stdin);
65             sscanf(buffer, "%hd %hd", &pkt.x, &pkt.y);
66             pkt.opcode = 1;
67             pthread_mutex_lock(&socket_mutex);
68             +ret;
69             pkt.turn = turn;
70             if ((ret = send(client_socket, &pkt, sizeof(network_packet), 0)) != sizeof(network_packet)) {
71                 perror("Send error");
72             } else {
73                 // printf("Sent: %hd %hd %hd %hd\n", pkt.player_num, pkt.opcode, pkt.x, pkt.y, pkt.ship_type);
74             }
75             if(totalNumOfPlayers != 2){
76                 if(turn == totalNumOfPlayers-1){
77                     turn = 0;
78                 }
79                 pthread_mutex_unlock(&socket_mutex);
80             } else {
81                 while (turn != player_number);
82             }
83         }
84     }
85 }
```

github-classroom[bot], 2 weeks ago · Initial commit

// network recv thread
86 void *rx(void *arg) {
87 pthread_detach(pthread_self());
88 int ret;
89 char buffer[BUFFER_SIZE];
90 network_packet pkt;
91 network_packet sendBackPkt;
92
93 // loop forever
94 }

Help us improve our support for C++ Take Short Survey Remind Me Later

github-classroom[bot], 2 weeks ago Ln 85, Col 2 Spaces: 4 UTF-8 LF ⌂ C++ Go Live ⌂ Quokka ⌂ Spell Linux ⌂ tabnine starter ⌂ See Tabnine Insights ⌂ prettier ⌂

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Path:** client.cpp - battleship-Simon-Biamo - Visual Studio Code
- Code Editor:** The main pane displays the C++ code for the client. The code handles socket communication, checks for player responses, and manages the ocean grid.
- Sidebar:** On the left, there's a sidebar with various icons representing file types and project components.
- Right Panel:** On the right, there's a panel showing a preview of the code or a related document.
- Bottom Status Bar:** Shows the current file (client.cpp), line (Ln 85, Col 2), and other status information.

```
95 // loop forever
96 while (1) {
97     // block on read
98     if ((ret = recv(client_socket, buffer, BUFFER_SIZE, 0)) == 0) {
99         perror("read failed");
100         exit(EXIT_FAILURE);
101     } else {
102         // check for valid packet length
103         if (ret != sizeof(network_packet)) {
104             printf("WARNING: packet size %d from client_socket ignored\n", ret);
105         } else {
106             // copy the receive buffer into the pkt struct
107             pthread_mutex_lock(&socket_mutex);
108             memcpy(&pkt, buffer, sizeof(network_packet));
109
110             // is this the player_num response sent after connect?
111             if (pkt.opcode == OPCODE_CONNECT) {
112                 if(!connected) {
113                     player_number = pkt.player_num;
114                     printf("You're player %d!\n", player_number);
115                     connected = true;
116                 } else {
117                     printf("Player %d connected!\n", pkt.player_num);
118                 }
119                 if(totalNumOfPlayers < pkt.numOfPlayers){
120                     totalNumOfPlayers = pkt.numOfPlayers;
121                 }
122                 // printf("Total players = %d\n", totalNumOfPlayers);
123             } else if(pkt.opcode == OPCODE_SHOT){
124                 sendBackPkt = theOceanGrid->handleShot(&pkt, &shipsRemaining);
125
126                 printf("OCEAN GRID\n");
127                 theOceanGrid->printGrid();
128                 if(shipsRemaining == 0) {
129                     printf("ALL PLAYER SHIPS HAVE SUNK!\n");
130                     printf("YOU LOSE!\n");
131                     sleep(1);
132                     printf("YOU LOSE!\n");
133                     sleep(1);
134                     printf("YOU LOSE!\n");
135                     sleep(3);
136                     printf("YOU LOSE!\n");
137                     printf("\033[0m");
138                     exit(0);
139                 }
140             }
141         }
142         turn = pkt.turn;
143     }
144 }
```

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities < Visual Studio Code. Date: Dec 18 03:48.
- File Explorer:** Shows files: grid.h, client.cpp, and tx.old*.
- Code Editor:** Displays the `client.cpp` file content. The code handles network communication between clients and a server, specifically managing turns and ship sinking logic.
- Status Bar:** Shows the file path as `github-classroom/bot`, a timestamp of 2 weeks ago, line numbers (Ln 85, Col 2), spaces (Spaces: 4), encoding (UTF-8), file type (C++), Go Live, Quokka, Spell, Linux, tabline status, Tabnine Insights, and Prettier.
- Bottom Status Bar:** Includes icons for GitHub, Stack Overflow, and Live Share.

```
grid.h M client.cpp x tx.old*
142     turn = pkt.turn;
143     sendBackPkt.turn = pkt.turn;
144     if(totalNumOfPlayers == 2){
145         if((player_number == 1) {
146             turn = 1;
147             sendBackPkt.turn = 1;
148         } else {
149             turn = 0;
150             sendBackPkt.turn = 0;
151         }
152     }
153     send(client_socket, &sendBackPkt, sizeof(network_packet), 0);
154     if(turn != player_number) {
155         sleep(6);
156         printf("Waiting for turn...\n");
157     }
158 } else {
159     if(pkt.turn - 1 == player_number || ((pkt.turn == 0 && player_number == 1) && totalNumOfPlayers)) {
160         theTargetGrid->handleResponse(&pkt, &shipsSunk);
161
162         printf("TARGET GRID\n");
163         theTargetGrid->printGrid();
164         if(shipSunk == 5) {
165             printf("ALL OPPONENTS SHIPS HAVE SUNK!\n");
166             printf("\x033[32mYOU WIN!\n");
167             printf("\x033[30m");
168             exit(0);
169         }
170         printf("Waiting for turn...\n");
171         fflush(stdout);
172     }
173 }
174 // display Received packet fields
175 // printf("\nReceived: %hhnd %hhnd %hhnd %hhnd\n", pkt.player_num, pkt.opcode, pkt.x, pkt.y, pkt.ship_type);
176
177 // repeat prompt
178 }
179 pthread_mutex_unlock(&socket_mutex);
180
181 }
182
183 return 0;
184 }

int main(int argc, char *argv[]) {
185     printf("Start Game or Exit Program\n");
186     printf("1. Start Game\n");
187     printf("2. Exit\n");
188
189     if(argc != 2) {
190         printf("Usage: %s [game|exit]\n", argv[0]);
191         exit(1);
192     }
193 }
```

A screenshot of Visual Studio Code showing the file `client.cpp`. The code is a C++ program that handles user input for starting a game or exiting. It uses `scanf` to read a character from the user. If the character is '1', it starts a game. If it's '2', it exits. Otherwise, it prints an error message. The code then initializes socket structures and connects to a server at `localhost` on port `5000`. It creates two threads to handle the connection. The code ends with a return statement.

```
187 printf("Start Game or Exit Program\n");
188 printf("1. Start Game\n");
189 printf("2. Exit\n");
190 printf("Selection: ");
191
192 int opt = 0;
193 while (opt != 1 || opt != 2)
194 {
195     scanf("%d", &opt);
196     if(opt == 1 || opt == 2){
197         break;
198     }
199     printf("1. Start Game\n");
200     printf("2. Exit");
201     printf("Selection: ");
202 }
203 if(opt == 2){
204     exit(0);
205 }
206
207 struct sockaddr_in serv_addr;
208 char const *server_address;
209 theOceanGrid = new oceanGrid();
210 theTargetGrid = new targetGrid();
211 theOceanGrid->configGrid();
212 pthread_mutex_init(&socket_mutex, NULL);
213
214 // use server addr if specified on command line, otherwise use localhost
215 if (argc > 1) {
216     server_address = argv[1];
217 } else {
218     server_address = LOCALHOST;
219 }
220
221 // create client socket
222 if ((client_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
223     perror("socket failed");
224     exit(EXIT_FAILURE);
225 }
226
227 // initialize sockaddr_in fields
228 serv_addr.sin_family = AF_INET;
229 serv_addr.sin_port = htons(SERVER_PORT);
230 if (inet_ntop(AF_INET, server_address, &serv_addr.sin_addr) == 0) {
231     perror("inet_ntop failed");
232     exit(EXIT_FAILURE);
233 }
234
235 // connect to server
236 if (connect(client_socket, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
237     perror("connect failed");
238     exit(EXIT_FAILURE);
239 }
240
241 // Create two threads
242 pthread_t thread0, thread1;
243 pthread_create(&thread0, NULL, tx, NULL);
244 pthread_create(&thread1, NULL, tx, NULL);
245
246 // Wait for threads to finish
247 pthread_join(thread0, NULL);
248 pthread_join(thread1, NULL);
249
250 return 0;
251 }
```

A screenshot of Visual Studio Code showing the same `client.cpp` code as the previous screenshot. A tooltip box is overlaid on the screen, containing the text "Screenshot captured" and "You can paste the image from the clipboard." This indicates that a screenshot has been taken of the code editor.

8. Output

The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal displays a session of a battleship game. The user has run the server and is now interacting with the client. The client prompts the user to place ships, specifically asking for the placement of a carrier.

```
phoenix-blamo@daphoenix:~/Documents/battleship-Simon-Blamo$ ./server
Server listening: socket fd 3, ip 0.0.0.0, port 8888
phoenix-blamo@daphoenix:~/Documents/battleship-Simon-Blamo$ ./Client
Start Game or Exit Program
1. Start Game
2. Exit
Selection: 1

Welcome to Battleship! Let's place your pieces to start the game!
Place your Carrier!

Enter the ship's X, Y coordinates, along with it's placement! Use H to p
lace the piece horizontally, and user V to place the ship vertically! (e
.g. 1 2 H):
```

The screenshot shows a Visual Studio Code interface with two terminal panes. The left terminal shows the server side of the battleship game, and the right terminal shows the client side.

Terminal 1 (Server):

```
phoenix-blamo@daphoenix:~/Documents/battleship-Simon-Blamo$ ./server
Server listening: socket fd 3, ip 0.0.0.0, port 8888
```

Terminal 2 (Client):

```
phoenix-blamo@daphoenix:~/Documents/battleship-Simon-Blamo$ ./client
Start Game or Exit Program
1. Start Game
2. Exit
Selection: 1

Welcome to Battleship! Let's place your pieces to start the game!
Place your Carrier!
```

Terminal 3 (Client):

```
Welcome to Battleship! Let's place your pieces to start the game!
Place your Carrier!
```

Terminal 4 (Client):

```
Enter the ship's X, Y coordinates, along with it's placement! Use H to p
lace the piece horizontally, and user V to place the ship vertically! (e
.g. 1 2 H): 1 1 V
C
C
C
C
C
```

Terminal 5 (Client):

```
Place your Battleship!
```

Terminal 6 (Client):

```
Enter the ship's X, Y coordinates, along with it's placement!: [REDACTED]
```


