Московский авиационный институт
(национальный исследовательский университет)

Курсовой проект
По курсу «Операционные системы»
Тема: Проектирование консольной клиент-серверной игры
«Морской бой»

Выполнила
студентка группы 8О-208Б
Понагайбо А.О.
Преподаватель:
Миронов Е.С.

Цель курсового проекта:

1. Приобретение практических навыков в использовании знаний, полученных в течении курса
2. Проведение исследования в выбранной предметной области

Задание: Необходимо спроектировать и реализовать консольную клиент-серверную игру «Морской бой» на основе сервера очередей ZeroMQ.

Основная идея:

Поля с кораблями хранятся как двумерные массивы. Игроки расставляют корабли (случайно или с помощью ввода координат), после чего поля отправляются серверу и хранятся там. Игроки вводят команды, которые обрабатываются сервером, после чего измененные поля отсылаются обратно игрокам и выводятся на экран. При этом каждый раз проверяется условие завершения игры (остались ли еще корабли у игроков). Если был убит последний корабль у какого-либо игрока, обоим игрокам отправляются сообщения об их победе или поражении, после чего игра завершается.

Текст программы:

makefile

```
CC = g++
CFLAGS = -std=c++11

all: server client

server: thrServer.o
	$(CC) -o server thrServer.o -lpthread -L/usr/local/lib -lzmq $(CFLAGS)
client: thrClient.o
	$(CC) -o client thrClient.o -L/usr/local/lib -lzmq $(CFLAGS)

thrServer.o: thrServer.c
	$(CC) -c thrServer.c
thrClient.o: thrClient.c
	$(CC) -c thrClient.c
clean:
	rm -f *.o server client
```

thrClient.c

```
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
#include <stdio_ext.h>
#include "zmq.h"

#define SEED (unsigned int)(time(NULL)*(rand()%10000))
#define MESSAGE_SIZE 256
char** myArea;
char** enemyArea;

char* playerID;
char* adr;

int CheckNear(int a, int b) {
    if (myArea[a][b] == 'o') return 0;
    if (a != 9) {
        if (myArea[a+1][b] == 'o') return 0;
        if (b != 0) {
            if (myArea[a][b-1] == 'o' || myArea[a+1][b-1] == 'o') return 0;
        }
        if (b != 9) {
            if (myArea[a][b+1] == 'o' || myArea[a+1][b+1] == 'o') return 0;
        }
    }
```

```
    if (a != 0) {
        if (myArea[a-1][b] == 'o') return 0;
        if (b != 9) {
            if (myArea[a-1][b+1] == 'o' || myArea[a][b+1] == 'o') return 0;
        }
        if (b != 0) {
            if (myArea[a-1][b-1] == 'o' || myArea[a][b-1] == 'o') return 0;
        }
    }
    return 1;
}

void CopyPrint(char** area, char* from, int copy) {
    int i, j;
    printf("\n  ");
    for (i = 0; i < 10; i++) printf("%d ", i);
    printf("\n");
    for (i = 0; i < 10; i++) {
        printf("%d ", i);
        for (j = 0; j < 10; j++) {
            if (copy) area[i][j] = from[i * 10 + j];
            printf("%c ", area[i][j]);
        }
        printf("\n");
    }
}

void RandomSet(const int flag) {
    int k, a, b, i, temp, r;
    for (k = 4; k > 0; k--) {
        int amount = 5 - k;
        while (amount) {
            if (flag) {
                srand(SEED);
                a = rand() % 10;
                b = rand() % 10;
                r = rand();
            } else {
                __fpurge(stdin);
                printf("Still have %d %d-decker ships to set\n", amount, k);
                printf("Enter orientation (1 - vertical, 0 -
horizontal)\n>");
                scanf("%1d", &r);
                printf("Enter coordinats of first ship decker:\n>");
                __fpurge(stdin);
                scanf("%1d", &a);
                __fpurge(stdin);
                scanf("%1d", &b);
            }
            temp = 0;
            if ((r % 2) == 0 && b + k - 1 <= 9) {
                for (i = 0; i < k; i++) {
                    if (CheckNear(a, b + i)) temp++;
                    else break;
                }
                if (temp == k) {
                    for (i = 0; i < k; i++) myArea[a][b + i] = 'o';
                    amount--;
                }
            } else if ((r % 2) != 0 && a + k - 1 <= 9) {
                for (i = 0; i < k; i++) {
                    if (CheckNear(a + i, b)) temp++;
                    else break;
                }
                if (temp == k) {
                    for (i = 0; i < k; i++) myArea[a + i][b] = 'o';
                    amount--;
                }
            } else if (!flag) printf("Wrong coordinats, try again\n");
```

```
            if (!flag) CopyPrint(myArea, 0, 0);
        }
    }
}

void Init() {
    int i, j;
    adr = (char*)malloc(32 * sizeof(char));
    myArea = (char**)malloc(10 * sizeof(char*));
    enemyArea = (char**)malloc(10 * sizeof(char*));
    for (i = 0; i < 10; i++) {
        myArea[i] = (char*)malloc(11 * sizeof(char));
        enemyArea[i] = (char*)malloc(11 * sizeof(char));
    }
    for (i = 0; i < 10; i++) {
        memset(myArea[i], 0, 11);
        memset(enemyArea[i], 0, 11);
        for (j = 0; j < 11; j++) {
            myArea[i][j] = '~';
            enemyArea[i][j] = '~';
        }
    }
}

void Free() {
    int i;
    for (i = 0; i < 10; i++) {
        free(myArea[i]);
        free(enemyArea[i]);
    }
    free(myArea);
    free(enemyArea);
    free(adr);
}

int Send(void* socket, char* ans) {
    if (zmq_send(socket, ans, MESSAGE_SIZE, 0) == -1) {
        printf("Error! Can not send message.\n");
        Free();
        return -1;
    } else return 0;
}

int Receive(void* socket, char* msg) {
    if (zmq_recv(socket, msg, MESSAGE_SIZE, 0) == -1) {
        printf("Error! Can not receive message.\n");
        Free();
        return -1;
    } else return 0;
}

int main(int argc, char* argv[]) {
    if (argc != 2) {
        printf("Error! You should set your ID (> 0) and port number (>
1024).\n");
        return -1;
    }
    int tempID;
    tempID = atoi(argv[1]);
    if (tempID == 0) {
        printf("Wrong ID! Try again.\n");
        return -1;
    }
    char* mainAdr;
    strcpy(mainAdr, "tcp://localhost:1024");
    void* mainContext = zmq_ctx_new();
    if (mainContext == NULL) {
        printf("Can not create mainContext.\n");
        return -1;
```

```
    }
    void* mainSocket = zmq_socket(mainContext, ZMQ_REQ);
    if (mainSocket == NULL) {
        printf("Can not create socket (mainSocket).\n");
        return -1;
    }
    if (zmq_connect(mainSocket, mainAdr) == -1) {
        printf("Can not connect.\n");
        return -1;
    }
    char tmp[MESSAGE_SIZE];
    char msg[MESSAGE_SIZE];
    char portNum[6];
    strcpy(tmp, "port");
    if (Send(mainSocket, tmp)) return -1;
    memset(tmp, 0, MESSAGE_SIZE);
    if (Receive(mainSocket, tmp)) return -1;
    sscanf(tmp, "%s %s", msg, portNum);
    if (!strcmp(msg, "error")) {
        printf("Error!\n");
        return -1;
    }
    zmq_close(mainSocket);
    zmq_ctx_destroy(mainContext);
    Init();
    strcpy(adr, "tcp://localhost:");
    strcat(adr, portNum);
    void* context = zmq_ctx_new();
    if (context == NULL) {
        printf("Can not create context.\n");
        return -1;
    }
    void* request = zmq_socket(context, ZMQ_REQ);
    if (request == NULL) {
        printf("Can not create socket (request).\n");
        return -1;
    }
    if (zmq_connect(request, adr) == -1) {
        printf("Can not connect.\n");
        return -1;
    }
    printf("Do you want set ships yourself (0) or randomly (1)?\n>");
    int randomly = 0;
    scanf("%1d", &randomly);
    RandomSet(randomly);
    int i, j;
    char OK[3] = "OK";
    strcpy(tmp, "");
    strcat(tmp, argv[1]);
    sprintf(msg, " area ");
    strcat(tmp, msg);
    char tempChar[11];
    for (i = 0; i < 10; i++) {
        memset(tempChar, 0, 11);
        for (j = 0; j < 10; j++) tempChar[j] = myArea[i][j];
        strcat(tmp, tempChar);
    }
    if (Send(request, tmp)) return -1;
    memset(tmp, 0, MESSAGE_SIZE);
    if (Receive(request, tmp)) return -1;
    printf("%s\n", tmp);
    if (!strcmp(tmp, "You have the first turn") || !strcmp(tmp, "You have
the second turn")) {
        if (Send(request, OK)) return -1;
    }
    char enem[MESSAGE_SIZE];
    char tempAns[MESSAGE_SIZE];
    char check[10];
    while (!feof(stdin)) {
```

```
        memset(tmp, 0, MESSAGE_SIZE);
        strcpy(tmp, "");
        if (Receive(request, tmp)) return -1;
        memset(enem, 0, MESSAGE_SIZE);
        sscanf(tmp, "%s %s", tempAns, enem);
        printf("%s\n", tempAns);
        if (!strcmp(tempAns, "wait")) {
            printf("Turn of another player:\n");
            printf("\tYOUR AREA");
            CopyPrint(myArea, enem, 1);
        } else if (!strcmp(tempAns, "your_turn")) {
            printf("\tYOUR AREA");
            CopyPrint(myArea, enem, 1);
            printf("Your turn:\n");
            printf("\tENEMY");
            CopyPrint(enemyArea, enem, 0);
        } else if (!strcmp(tempAns, "killed") || !strcmp(tempAns,
"injured")) {
            printf("\tYOUR AREA");
            CopyPrint(myArea, enem, 0);
            printf("Your turn:\n");
            printf("\tENEMY");
            CopyPrint(enemyArea, enem, 1);
        } else if (!strcmp(tempAns, "miss")) {
            printf("\tENEMY");
            CopyPrint(enemyArea, enem, 1);
            printf("Turn of another player:\n");
        } else if (!strcmp(tempAns, "marked")) printf("Choose another
cell\n");
        if (!strcmp(tempAns, "your_turn") || !strcmp(tempAns, "killed") || !
strcmp(tempAns, "injured") || !strcmp(tempAns, "marked")) {
            while (1) {
                __fpurge(stdin);
                printf("\nEnter your command:\n>");
                fgets(msg, MESSAGE_SIZE, stdin);
                int tmpL = 0;
                int tmpC = 0;
                strcpy(tmp, msg);
                sscanf(msg, "%s %d %d", check, &tmpL, &tmpC);
                if (strcmp(check, "shoot")) printf("Wrong command!\n");
                if (!strcmp(check, "shoot") && tmpL >= 0 && tmpL <= 9 &&
tmpC >= 0 && tmpC <= 9)
                    break;
                else printf("Wrong coordinates!\n");
            }
            if (Send(request, tmp)) return -1;
        } else if (!strcmp(tempAns, "miss") || !strcmp(tempAns, "wait")) {
            printf("Wait for your turn\n");
            if (Send(request, OK)) return -1;
        } else if (!strcmp(tempAns, "won")) {
            printf("You win!\n");
            if (Send(request, OK)) return -1;
            return 0;
        } else if (!strcmp(tempAns, "lost")) {
            printf("You lose!\n");
            if (Send(request, OK)) return -1;
            return 0;
        } else {
            printf("Unknown command\n");
            sprintf(tmp, "error");
            if (Send(request, OK)) return -1;
        }
    }
    zmq_close(request);
    zmq_ctx_destroy(context);
    Free();
    return 0;
}
```

thrServer.c

```
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>
#include <stdio_ext.h>
#include "zmq.h"

#define MESSAGE_SIZE 150

int mainID;
typedef struct {
    int ID;
    int _4d;
    int _3d;
    int _2d;
    int _1d;
} TPlayer;

int MarkNear(const int ln, const int cl, const int orient, const int deck, char**
area) {
    if (orient == 0) {
        if (ln == 0) {
            area[ln + 1][cl] = '*';
            if (deck == 1 && cl != 10 - deck) {
                area[ln + 1][cl + 1] = '*';
                area[ln][cl + 1] = '*';
            }
            if (cl != 0) {
                area[ln + 1][cl - 1] = '*';
                if (area[ln][cl - 1] != 'X') area[ln][cl - 1] = '*';
            }
        } else if (ln == 9) {
            area[ln - 1][cl] = '*';
            if (deck == 1 && cl != 10 - deck) {
                area[ln - 1][cl + 1] = '*';
                area[ln][cl + 1] = '*';
            }
            if (cl != 0) {
                area[ln - 1][cl - 1] = '*';
                if (area[ln][cl - 1] != 'X') area[ln][cl - 1] = '*';
            }
        } else {
            area[ln - 1][cl] = '*';
            area[ln + 1][cl] = '*';
            if (deck == 1 && cl != 10 - deck) {
                area[ln - 1][cl + 1] = '*';
                area[ln][cl + 1] = '*';
                area[ln + 1][cl + 1] = '*';
            }
            if (cl != 0) {
                area[ln - 1][cl - 1] = '*';
                area[ln + 1][cl - 1] = '*';
                if (area[ln][cl - 1] != 'X')
                    area[ln][cl - 1] = '*';
            }
        }
        if (deck > 1) MarkNear(ln, cl + 1, 0, deck - 1, area);
    }
    if (orient == 1) {
        if (cl == 0) {
            area[ln][cl + 1] = '*';
            if (ln == 0) {
                if (deck == 1) {
                    area[ln + 1][cl] = '*';
                    area[ln + 1][cl + 1] = '*';
                }
```

```
            } else {
                area[ln - 1][cl + 1] = '*';
                if (area[ln - 1][cl] != 'X') area[ln - 1][cl] = '*';
                if (deck == 1 && ln != 10 - deck) {
                    area[ln + 1][cl] = '*';
                    area[ln + 1][cl + 1] = '*';
                }
            }
        } else if (cl == 9) {
            area[ln][cl - 1] = '*';
            if (ln == 0) {
                if (deck == 1) {
                    area[ln + 1][cl] = '*';
                    area[ln + 1][cl - 1] = '*';
                }
            } else {
                area[ln - 1][cl - 1] = '*';
                if (area[ln - 1][cl] != 'X') area[ln - 1][cl] = '*';
                if (deck == 1 && ln != 10 - deck) {
                    area[ln + 1][cl] = '*';
                    area[ln + 1][cl - 1] = '*';
                }
            }
        } else {
            area[ln][cl + 1] = '*';
            area[ln][cl - 1] = '*';
            if (ln == 0) {
                if (deck == 1) {
                    area[ln + 1][cl - 1] = '*';
                    area[ln + 1][cl] = '*';
                    area[ln + 1][cl + 1] = '*';
                }
            } else {
                area[ln - 1][cl + 1] = '*';
                area[ln - 1][cl - 1] = '*';
                if (area[ln - 1][cl] != 'X') area[ln - 1][cl] = '*';
                if (deck == 1 && ln != 10 - deck) {
                    area[ln + 1][cl + 1] = '*';
                    area[ln + 1][cl] = '*';
                    area[ln + 1][cl - 1] = '*';
                }
            }
        }
        if (deck > 1) MarkNear(ln + 1, cl, 1, deck - 1, area);
    }
    return 1;
}

void ReduceShips(const int deck, TPlayer* player) {
    if (deck == 4) player->_4d--;
    else if (deck == 3) player->_3d--;
    else if (deck == 2) player->_2d--;
    else if (deck == 1) player->_1d--;
}

int CheckVictory(TPlayer* player1, TPlayer* player2) {
    if (player1->_4d == 0 && player1->_3d == 0 && player1->_2d == 0 && player1->_1d ==
0) return 2;
    if (player2->_4d == 0 && player2->_3d == 0 && player2->_2d == 0 && player2->_1d ==
0) return 1;
    else return 0;
}

int CheckIfKilled(const int line, const int column, char** area, TPlayer* player) {
    int hor = 0;
    int vert = 0;
    int inj = 0;
    int cnt = 1;
    while (cnt < 4) {
```

```
        if (line + cnt > 9) break;
        if (area[line + cnt][column] == 'X' || area[line + cnt][column] == 'o') {
            vert++;
            if (area[line + cnt][column] == 'X') inj++;
            cnt++;
        } else break;
    }
    cnt = 1;
    while (cnt < 4) {
        if (line - cnt < 0) break;
        if (area[line - cnt][column] == 'X' || area[line - cnt][column] == 'o') {
            vert++;
            if (area[line - cnt][column] == 'X') inj++;
            cnt++;
        } else break;
    }
    cnt = 1;
    while (cnt < 4) {
        if (column + cnt > 9) break;
        if (area[line][column + cnt] == 'X' || area[line][column + cnt] == 'o') {
            hor++;
            if (area[line][column + cnt] == 'X') inj++;
            cnt++;
        } else break;
    }
    cnt = 1;
    while (cnt < 4) {
        if (column - cnt < 0) break;
        if (area[line][column - cnt] == 'X' || area[line][column - cnt] == 'o') {
            hor++;
            if (area[line][column - cnt] == 'X') inj++;
            cnt++;
        } else break;
    }
    cnt = 1;
    int killed = 0;
    if (inj == vert + hor) killed = 1;
    int dc = 1 + vert + hor;
    int begL, begC;
    begC = column;
    begL = line;
    cnt = 1;
    if (vert != 0) {
        while (line - cnt >= 0 && (area[line - cnt][column] == 'X' || area[line - cnt]
[column] == 'o') ) {
            begL--;
            cnt++;
        }
    } else if (hor != 0) {
        while (column - cnt >= 0 && (area[line][column - cnt] == 'X' || area[line]
[column - cnt] == 'o')) {
            begC--;
            cnt++;
        }
    }
    if (killed) {
        if (hor != 0) MarkNear(begL, begC, 0, dc, area);
        else MarkNear(begL, begC, 1, dc, area);
        ReduceShips(dc, player);
        return 1;
    }
    return 0;
}

void PrintSecond(char** secondArea, int ID) {
    int i, j;
    printf("\tSECOND, ID: %d\n  ", ID);
    for (i = 0; i < 10; i++) printf("%d ", i);
    printf("\n");
```

```
    for (i = 0; i < 10; i++) {
        printf("%d ", i);
        for (j = 0; j < 10; j++) printf("%c ", secondArea[i][j]);
        printf("\n");
    }
}

void PrintFirst(char** firstArea, int ID) {
    int i, j;
    printf("\tFIRST, ID: %d\n  ", ID);
    for (i = 0; i < 10; i++) printf("%d ", i);
    printf("\n");
    for (i = 0; i < 10; i++) {
        printf("%d ", i);
        for (j = 0; j < 10; j++) printf("%c ", firstArea[i][j]);
        printf("\n");
    }
}

void Free(char** firstArea, char** secondArea, TPlayer* player1, TPlayer* player2,
char* adr1, char* adr2) {
    int i;
    for (i = 0; i < 10; i++) {
        free(firstArea[i]);
        free(secondArea[i]);
    }
    free(firstArea);
    free(secondArea);
    free(player1);
    free(player2);
    free(adr1);
    free(adr2);
}

void AddToAnswer(char** area, char* ans, const int hide) {
    char tempChar[11];
    int i, j;
    for (i = 0; i < 10; i++) {
        memset(tempChar, 0, 11);
        for (j = 0; j < 10; j++) {
            if (hide) {
                if (area[i][j] == 'o') tempChar[j] = '~';
                else tempChar[j] = area[i][j];
            } else tempChar[j] = area[i][j];
        }
        strcat(ans, tempChar);
    }
}

int Miss(char** area, void* socket1, void* socket2) {
    char ans[MESSAGE_SIZE];
    char cmd2[MESSAGE_SIZE];
    sprintf(ans, "miss ");
    AddToAnswer(area, ans, 1);
    if (zmq_send(socket1, ans, strlen(ans), 0) == -1) {
        printf("Error! Can not send message3.\n");
        return -1;
    }
    sprintf(cmd2, "your_turn ");
    AddToAnswer(area, cmd2, 0);
    if (zmq_send(socket2, cmd2, strlen(cmd2), 0) == -1) {
        printf("Error! Can not send message31.\n");
        return -1;
    }
}

int EndGame(void* socket1, void* socket2) {
    char ans[MESSAGE_SIZE];
    char tmp[MESSAGE_SIZE];
```

```
    sprintf(ans, "won ");
    if (zmq_send(socket1, ans, strlen(ans), 0) == -1) {
        printf("Error! Can not send message4.\n");
        return -1;
    }
    sprintf(ans, "lost ");
    if (zmq_send(socket2, ans, strlen(ans), 0) == -1) {
        printf("Error! Can not send message52.\n");
        return -1;
    }
    if (zmq_recv(socket1, tmp, MESSAGE_SIZE, 0) == -1) {
        printf("Error! Can not receive message9.\n");
        return -1;
    }
    if (zmq_recv(socket2, tmp, MESSAGE_SIZE, 0) == -1) {
        printf("Error! Can not receive message10.\n");
        return -1;
    }
    if (zmq_send(socket1, ans, strlen(ans), 0) == -1) {
        printf("Error! Can not send message4.\n");
        return -1;
    }
    if (zmq_send(socket2, ans, strlen(ans), 0) == -1) {
        printf("Error! Can not send message53.\n");
        return -1;
    }
}

void CopyPrint(char** area, char* from) {
    int i, j;
    printf("\n  ");
    for (i = 0; i < 10; i++) printf("%d ", i);
    printf("\n");
    for (i = 0; i < 10; i++) {
        printf("%d ", i);
        for (j = 0; j < 10; j++) {
            area[i][j] = from[i * 10 + j];
            printf("%c ", area[i][j]);
        }
        printf("\n");
    }
}

typedef struct {
    char* firstPort;
    char* secondPort;
} treadInfo;

int Send(void* socket, char* ans, char** firstArea, char** secondArea, TPlayer*
player1, TPlayer* player2, char* adr1, char* adr2) {
    if (zmq_send(socket, ans, MESSAGE_SIZE, 0) == -1) {
        printf("Error! Can not send message.\n");
        Free(firstArea, secondArea, player1, player2, adr1, adr2);
        return -1;
    } else return 0;
}

int Receive(void* socket, char* msg, char** firstArea, char** secondArea, TPlayer*
player1, TPlayer* player2, char* adr1, char* adr2) {
    if (zmq_recv(socket, msg, MESSAGE_SIZE, 0) == -1) {
        printf("Error! Can not receive message.\n");
        Free(firstArea, secondArea, player1, player2, adr1, adr2);
        return -1;
    } else return 0;
}

int StartGame(char* firstAdr, char* secondAdr) {
    char** area1;
    char** area2;
```

```
char* adr1;
char* adr2;
TPlayer* pl1;
TPlayer* pl2;
int i,j;
void* context1 = zmq_ctx_new();
if (context1 == NULL) {
    printf("Can not create context.\n");
    return -1;
}
void* context2 = zmq_ctx_new();
if (context2 == NULL) {
    printf("Can not create context.\n");
    return -1;
}
void* respond1 = zmq_socket(context1, ZMQ_REP);
if (respond1 == NULL) {
    printf("Can not create socket (responder).\n");
    return -1;
}
void* respond2 = zmq_socket(context1, ZMQ_REP);
if (respond2 == NULL) {
    printf("Can not create socket (responder).\n");
    return -1;
}
adr1 = (char*)malloc(32 * sizeof(char));
adr2 = (char*)malloc(32 * sizeof(char));
area1 = (char**)malloc(10 * sizeof(char*));
area2 = (char**)malloc(10 * sizeof(char*));
for (i = 0; i < 10; i++) {
    area1[i] = (char*)malloc(11 * sizeof(char));
    area2[i] = (char*)malloc(11 * sizeof(char));
}
pl1 = (TPlayer*)malloc(sizeof(TPlayer));
pl2 = (TPlayer*)malloc(sizeof(TPlayer));
pl1->ID = 0;
pl2->ID = 0;
pl1->_1d = 4;
pl1->_2d = 3;
pl1->_3d = 2;
pl1->_4d = 1;
pl2->_1d = 4;
pl2->_2d = 3;
pl2->_3d = 2;
pl2->_4d = 1;
strcpy(adr1, "tcp://*:");
strcat(adr1, firstAdr);
strcpy(adr2, "tcp://*:");
strcat(adr2, secondAdr);
if (zmq_bind(respond1, adr1) == -1) {
    printf("Can not bind socket1.\n");
    Free(area1, area2, pl1, pl2, adr1, adr2);
    return -1;
}
if (zmq_bind(respond2, adr2) == -1) {
    printf("Can not bind socket2.\n");
    Free(area1, area2, pl1, pl2, adr1, adr2);
    return -1;
}
char tmp[MESSAGE_SIZE];
char cmd[MESSAGE_SIZE];
strcpy(cmd, "");
strcpy(tmp, "");
if (Receive(respond1, tmp, area1, area2, pl1, pl2, adr1, adr2)) return -1;
sscanf(tmp, "%d %s", &pl1->ID, cmd);
if (!strcmp(cmd, "area")) {
    sscanf(tmp, "%*d %*s %s", tmp);
    printf("\tFIRST PLAYER");
    CopyPrint(area1, tmp);
```

```
    }
    char ans[MESSAGE_SIZE];
    sprintf(ans, "You have the first turn");
    if (Send(respond1, ans, area1, area2, pl1, pl2, adr1, adr2)) return -1;
    strcpy(cmd, "");
    strcpy(tmp, "");
    memset(tmp, 0, MESSAGE_SIZE);
    if (Receive(respond2, tmp, area1, area2, pl1, pl2, adr1, adr2)) return -1;
    sscanf(tmp, "%d %s", &pl2->ID, cmd);
    if (!strcmp(cmd, "area")) {
        sscanf(tmp, "%*d %*s %s", tmp);
        printf("\tSECOND PLAYER");
        CopyPrint(area2, tmp);
    }
    sprintf(ans, "You have the second turn");
    if (Send(respond2, ans, area1, area2, pl1, pl2, adr1, adr2)) return -1;
    char tmp2[MESSAGE_SIZE];
    char cmd2[MESSAGE_SIZE];
    char tempChar[11];
    strcpy(cmd, "");
    strcpy(tmp, "");
    if (Receive(respond1, tmp, area1, area2, pl1, pl2, adr1, adr2)) return -1;
    if (Receive(respond2, tmp, area1, area2, pl1, pl2, adr1, adr2)) return -1;
    sprintf(cmd2, "your_turn ");
    AddToAnswer(area1, cmd2, 0);
    if (Send(respond1, cmd2, area1, area2, pl1, pl2, adr1, adr2)) return -1;
    sprintf(cmd2, "wait ");
    AddToAnswer(area2, cmd2, 0);
    if (Send(respond2, cmd2, area1, area2, pl1, pl2, adr1, adr2)) return -1;
    while (1) {
        strcpy(cmd, "");
        strcpy(tmp, "");
        if (Receive(respond1, tmp, area1, area2, pl1, pl2, adr1, adr2)) return -1;
        if (Receive(respond2, tmp2, area1, area2, pl1, pl2, adr1, adr2)) return -1;
        sscanf(tmp, "%s", cmd);
        while (!strcmp(cmd, "shoot")) {
            int l, col;
            sscanf(tmp, "%*s %d %d", &l, &col);
            if (area2[l][col] == '~') {
                area2[l][col] = '*';
                if (Miss(area2, respond1, respond2)) {
                    Free(area1, area2, pl1, pl2, adr1, adr2);
                    return -1;
                }
                break;
            } else if (area2[l][col] == 'o') {
                area2[l][col] = 'X';
                if (CheckIfKilled(l, col, area2, pl2)) {
                    if (CheckVictory(pl1, pl2) == 1) {
                        printf("first player win\n");
                        if (EndGame(respond1, respond2)) {
                            Free(area1, area2, pl1, pl2, adr1, adr2);
                            return -1;
                        }
                        return 0;
                    }
                    sprintf(ans, "killed ");
                } else sprintf(ans, "injured ");
                AddToAnswer(area2, ans, 1);
                if (Send(respond1, ans, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                sprintf(cmd2, "wait ");
                AddToAnswer(area2, cmd2, 0);
                if (Send(respond2, cmd2, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                PrintSecond(area2, pl2->ID);
                strcpy(tmp, "");
                if (Receive(respond1, tmp, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
```

```
                if (Receive(respond2, tmp2, area1, area2, pl1, pl2, adr1, adr2))
return -1;
            } else {
                sprintf(ans, "marked ");
                if (Send(respond1, ans, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                sprintf(cmd2, "wait ");
                AddToAnswer(area2, cmd2, 0);
                if (Send(respond2, cmd2, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                strcpy(tmp, "");
                if (Receive(respond1, tmp, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                if (Receive(respond2, tmp2, area1, area2, pl1, pl2, adr1, adr2))
return -1;
            }
        }
        PrintSecond(area2, pl2->ID);
        strcpy(tmp2, "");
        if (Receive(respond1, tmp, area1, area2, pl1, pl2, adr1, adr2)) return -1;
        if (Receive(respond2, tmp2, area1, area2, pl1, pl2, adr1, adr2)) return -1;
        while (!strcmp(cmd, "shoot")) {
            int l, col;
            sscanf(tmp2, "%*s %d %d", &l, &col);
            if (area1[l][col] == '~') {
                area1[l][col] = '*';
                if (Miss(area1, respond2, respond1)) {
                    Free(area1, area2, pl1, pl2, adr1, adr2);
                    return -1;
                }
                break;
            } else if (area1[l][col] == 'o') {
                area1[l][col] = 'X';
                if (CheckIfKilled(l, col, area1, pl1)) {
                    if (CheckVictory(pl1, pl2) == 2) {
                        printf("second player win\n");
                        if (EndGame(respond2, respond1)) {
                            Free(area1, area2, pl1, pl2, adr1, adr2);
                            return -1;
                        }
                        return 0;
                    }
                    sprintf(ans, "killed ");
                } else sprintf(ans, "injured ");
                AddToAnswer(area1, ans, 1);
                if (Send(respond2, ans, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                sprintf(cmd2, "wait ");
                AddToAnswer(area1, cmd2, 0);
                if (Send(respond1, cmd2, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                PrintFirst(area1, pl1->ID);
                if (Receive(respond1, tmp, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                if (Receive(respond2, tmp2, area1, area2, pl1, pl2, adr1, adr2))
return -1;
            } else {
                sprintf(ans, "marked ");
                sprintf(cmd2, "wait ");
                AddToAnswer(area1, cmd2, 0);
                if (Send(respond1, cmd2, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                if (Send(respond2, ans, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                strcpy(tmp2, "");
                if (Receive(respond1, tmp, area1, area2, pl1, pl2, adr1, adr2)) return
-1;
                if (Receive(respond2, tmp2, area1, area2, pl1, pl2, adr1, adr2))
return -1;
```

```
            }
        }
        PrintFirst(area1, pl1->ID);
    }
    zmq_close(respond1);
    zmq_close(respond2);
    zmq_ctx_destroy(context1);
    zmq_ctx_destroy(context2);
    Free(area1, area2, pl1, pl2, adr1, adr2);
    return 0;
}


void* SendGame(void* data) {
    treadInfo* ndata = (treadInfo*)data;
    if (StartGame(ndata->firstPort, ndata->secondPort)) printf("Error!\n");
}

int main() {
    int dataSize = 1;
    treadInfo* data = (treadInfo*)malloc(dataSize * sizeof(treadInfo));
    pthread_t* threads = (pthread_t*)malloc(dataSize * sizeof(pthread_t));
    while (1) {
        void* mainContext1 = zmq_ctx_new();
        if (mainContext1 == NULL) {
            printf("Can not create context.\n");
            return -1;
        }
        void* mainSocket1 = zmq_socket(mainContext1, ZMQ_REP);
        if (mainSocket1 == NULL) {
            printf("Can not create socket (responder).\n");
            return -1;
        }
        char* mainAdr = (char*)malloc(32 * sizeof(char));
        strcpy(mainAdr, "tcp://*:1024");
        if (zmq_bind(mainSocket1, mainAdr) == -1) {
            printf("Can not bind socket1024.\n");
            free(mainAdr);
            return -1;
        }
        data = (treadInfo*)realloc(data, dataSize * sizeof(treadInfo));
        data[dataSize - 1].firstPort = (char*)malloc(MESSAGE_SIZE * sizeof(char));
        data[dataSize - 1].secondPort = (char*)malloc(MESSAGE_SIZE * sizeof(char));
        threads = (pthread_t*)realloc(threads, dataSize * sizeof(pthread_t));
        sprintf(data[dataSize - 1].firstPort, "%d", 1025 + 2 * dataSize - 1);
        sprintf(data[dataSize - 1].secondPort, "%d", 1025 + 2 * dataSize);

        char tmp[MESSAGE_SIZE];
        char ans[MESSAGE_SIZE];
        strcpy(ans, "");
        strcpy(tmp, "");
        if (zmq_recv(mainSocket1, tmp, MESSAGE_SIZE, 0) == -1) {
            printf("Error! Can not receive message001111.\n");
            free(mainAdr);
            return -1;
        }
        if (!strcmp(tmp, "port")) {
            sprintf(ans, "port %s", data[dataSize - 1].firstPort);
            if (zmq_send(mainSocket1, ans, strlen(ans), 0) == -1) {
                printf("Error! Can not send message1111.\n");
                free(mainAdr);
                return -1;
            }
        }
        sleep(1);
        if (zmq_recv(mainSocket1, tmp, MESSAGE_SIZE, 0) == -1) {
            printf("Error! Can not receive message991111.\n");
            free(mainAdr);
            return -1;
```

```
        }
        if (!strcmp(tmp, "port")) {
            sprintf(ans, "port %s", data[dataSize - 1].secondPort);
            if (zmq_send(mainSocket1, ans, strlen(ans), 0) == -1) {
                printf("Error! Can not send message12222.\n");
                free(mainAdr);
                return -1;
            }
        } else {
            printf("Error!\n");
            sprintf(ans, "error");
            if (zmq_send(mainSocket1, ans, strlen(ans), 0) == -1) {
                printf("Error! Can not send message1333.\n");
                free(mainAdr);
                return -1;
            }
            sleep(1);
            if (zmq_send(mainSocket1, ans, strlen(ans), 0) == -1) {
                printf("Error! Can not send message14444.\n");
                free(mainAdr);
                return -1;
            }
            free(mainAdr);
            return -1;
        }
        zmq_close(mainSocket1);
        zmq_ctx_destroy(mainContext1);
        free(mainAdr);
        int iret1 = pthread_create( &threads[dataSize - 1], NULL, SendGame,
&data[dataSize - 1]);
        if (iret1) {
            fprintf(stderr,"Error - pthread_create() return code: %d\nNum: %d\n",
iret1, dataSize);
            exit(EXIT_FAILURE);
        }
        dataSize++;
    }
    int i;
    for (i = 0; i < dataSize; i++) {
        free(data[i].firstPort);
        free(data[i].secondPort);
    }
    free(threads);
    free(data);
    return 0;
}
```

Тесты:

```
CourseWork/thread$ ./client
31
Do you want set ships
yourself (0) or randomly
(1)?
>1
You have the first turn
your_turn
      YOUR AREA
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ o ~ ~ o ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 o ~ o ~ ~ ~ ~ ~ ~ ~
3 o ~ o ~ ~ o o ~ ~ ~
4 o ~ o ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ o o o o
6 ~ o ~ o ~ ~ ~ ~ ~ ~
7 ~ ~ ~ o ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ o o ~
```

```
CourseWork/thread$ ./server
      FIRST PLAYER
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ o ~ ~ o ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 o ~ o ~ ~ ~ ~ ~ ~ ~
3 o ~ o ~ ~ o o ~ ~ ~
4 o ~ o ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ o o o o
6 ~ o ~ o ~ ~ ~ ~ ~ ~
7 ~ ~ ~ o ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ o o ~
9 ~ o ~ ~ ~ ~ ~ ~ ~ ~
      SECOND PLAYER
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1 ~ ~ ~ ~ ~ ~ o ~ ~ o
2 ~ ~ o ~ o ~ ~ ~ ~ ~
3 ~ ~ ~ ~ o ~ ~ ~ ~ o
4 o ~ ~ ~ ~ ~ ~ ~ ~ o
5 ~ ~ ~ ~ ~ o o ~ ~ o
6 ~ ~ o ~ ~ ~ ~ ~ ~ ~
7 ~ ~ o ~ ~ ~ ~ ~ ~ ~
8 o ~ o ~ ~ ~ o o o o
```

```
CourseWork/thread$ ./client
32
Do you want set ships
yourself (0) or randomly (1)?
>1
You have the second turn
wait
Turn of another player:
      YOUR AREA
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1 ~ ~ ~ ~ ~ ~ o ~ ~ o
2 ~ ~ o ~ o ~ ~ ~ ~ ~
3 ~ ~ ~ ~ o ~ ~ ~ ~ o
4 o ~ ~ ~ ~ ~ ~ ~ ~ o
5 ~ ~ ~ ~ ~ o o ~ ~ o
6 ~ ~ o ~ ~ ~ ~ ~ ~ ~
7 ~ ~ o ~ ~ ~ ~ ~ ~ ~
8 o ~ o ~ ~ ~ o o o o
```

```
9 ~ o ~ ~ ~ ~ ~ ~ ~ ~
Your turn:
      ENEMY
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
9 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

Enter your command:
>shoot 9 4
miss
      ENEMY
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
9 ~ ~ ~ ~ * ~ ~ ~ ~ ~
Turn of another player:
Wait for your turn
your_turn
      YOUR AREA
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ o ~ ~ o ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 o ~ o ~ ~ ~ ~ ~ ~ ~
3 o ~ o ~ ~ o o ~ ~ ~
4 o ~ o ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ o o o o
6 ~ o * o ~ ~ ~ ~ ~ ~
7 ~ ~ ~ o ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ o o ~
9 ~ o ~ ~ ~ ~ ~ ~ ~ ~
Your turn:
      ENEMY
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
9 ~ ~ ~ ~ * ~ ~ ~ ~ ~

Enter your command:
>shoot 1 6
killed
      YOUR AREA
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ o ~ ~ o ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 o ~ o ~ ~ ~ ~ ~ ~ ~
3 o ~ o ~ ~ o o ~ ~ ~
4 o ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ o o o o
6 ~ o * o ~ ~ ~ ~ ~ ~
7 ~ ~ ~ o ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ o o ~
9 ~ o ~ ~ ~ ~ ~ ~ ~ ~
      SECOND, ID: 32
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ o ~ o ~ ~ ~ ~ ~
3 ~ ~ ~ ~ o ~ ~ ~ ~ o
4 o ~ ~ ~ ~ ~ ~ ~ ~ o
5 ~ ~ ~ ~ ~ o o ~ ~ o
6 ~ ~ o ~ ~ ~ ~ ~ ~ ~
7 ~ ~ o ~ ~ ~ ~ ~ ~ ~
8 o ~ o ~ ~ ~ o o o o
9 o ~ ~ ~ * ~ ~ ~ ~ ~
      FIRST, ID: 31
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ o ~ ~ ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 o ~ o ~ ~ ~ ~ ~ ~ ~
3 o ~ o ~ ~ o o ~ ~ ~
4 o ~ o ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ o o o o
6 ~ o * o ~ ~ ~ ~ ~ ~
7 ~ ~ ~ o ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ o o ~
9 ~ o ~ ~ ~ ~ ~ ~ ~ ~
      SECOND, ID: 32
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ * * * ~ ~
1 ~ ~ ~ ~ ~ ~ * X * ~ o
2 ~ ~ o ~ o * * * ~ ~
3 ~ ~ ~ ~ o ~ ~ ~ ~ o
4 o ~ ~ ~ ~ ~ ~ ~ ~ o
5 ~ ~ ~ ~ ~ ~ X o ~ ~ o
6 ~ ~ o ~ ~ ~ ~ ~ ~ ~
7 ~ ~ o ~ ~ ~ ~ ~ ~ ~
8 o ~ o ~ ~ ~ o o o o
9 o ~ ~ ~ * ~ ~ ~ ~ ~
      SECOND, ID: 32
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ * * * ~ ~
1 ~ ~ ~ ~ ~ ~ * X * ~ o
2 ~ ~ o ~ o * * * ~ ~
3 ~ ~ ~ ~ o ~ ~ ~ ~ o
4 o ~ ~ ~ ~ ~ ~ ~ ~ o
5 ~ ~ ~ ~ ~ X o ~ ~ o
6 ~ ~ o ~ ~ * ~ ~ ~ ~
7 ~ ~ o ~ ~ ~ ~ ~ ~ ~
8 o ~ o ~ ~ ~ o o o o
9 o ~ ~ ~ * ~ ~ ~ ~ ~
      FIRST, ID: 31
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
9 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

Enter your command:
>shoot 6 2
miss
      ENEMY
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ * ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
9 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
Turn of another player:
Wait for your turn
wait
Turn of another player:
      YOUR AREA
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ * * * ~ ~
1 ~ ~ ~ ~ ~ ~ * X * ~ o
2 ~ ~ o ~ o * * * ~ ~
3 ~ ~ ~ ~ o ~ ~ ~ ~ o
4 o ~ ~ ~ ~ ~ ~ ~ ~ o
5 ~ ~ ~ ~ ~ o o ~ ~ o
6 ~ ~ o ~ ~ ~ ~ ~ ~ ~
7 ~ ~ o ~ ~ ~ ~ ~ ~ ~
8 o ~ o ~ ~ ~ o o o o
9 o ~ ~ ~ * ~ ~ ~ ~ ~
Wait for your turn
wait
Turn of another player:
      YOUR AREA
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ * * * ~ ~
1 ~ ~ ~ ~ ~ ~ * X * ~ o
2 ~ ~ o ~ o * * * ~ ~
```

```
4 o ~ o ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ o o o o
6 ~ o * o ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ o ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ o o ~
9 ~ o ~ ~ ~ ~ ~ ~ ~ ~ ~
Your turn:
      ENEMY
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ * * * ~ ~
1 ~ ~ ~ ~ ~ ~ * X * ~ ~
2 ~ ~ ~ ~ ~ ~ * * * ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
9 ~ ~ ~ ~ * ~ ~ ~ ~ ~ ~

Enter your command:
>shoot 1 7
marked
Choose another cell

Enter your command:
>shoot 5 5
injured
      YOUR AREA
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ o ~ ~ o ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 o ~ o ~ ~ ~ ~ ~ ~ ~ ~
3 o ~ o ~ ~ ~ o o ~ ~ ~
4 o ~ o ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ o o o o
6 ~ o * o ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ o ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ o o ~
9 ~ o ~ ~ ~ ~ ~ ~ ~ ~ ~
Your turn:
      ENEMY
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ * * * ~ ~
1 ~ ~ ~ ~ ~ ~ * X * ~ ~
2 ~ ~ ~ ~ ~ ~ * * * ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ X ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
9 ~ ~ ~ ~ * ~ ~ ~ ~ ~ ~

Enter your command:
>shoot 6 5
miss
      ENEMY
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ * * * ~ ~
1 ~ ~ ~ ~ ~ ~ * X * ~ ~
2 ~ ~ ~ ~ ~ ~ * * * ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ X ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ * ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
```

```
0 ~ ~ ~ ~ ~ ~ o ~ ~ o ~
1 ~ * ~ ~ ~ ~ ~ ~ ~ ~ ~
2 o ~ ~ o ~ ~ ~ ~ ~ ~ ~
3 o ~ ~ o ~ ~ ~ o o ~ ~ ~
4 o ~ ~ o ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ o o o o
6 ~ o * o ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ o ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ o o ~
9 ~ o ~ ~ ~ ~ ~ ~ ~ ~ ~
```

```
3 ~ ~ ~ ~ ~ o ~ ~ ~ ~ o
4 o ~ ~ ~ ~ ~ ~ ~ ~ ~ o
5 ~ ~ ~ ~ ~ ~ o o ~ ~ o
6 ~ ~ o ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ o ~ ~ ~ ~ ~ ~ ~ ~
8 o ~ o ~ ~ ~ ~ o o o o
9 o ~ ~ ~ * ~ ~ ~ ~ ~ ~
Wait for your turn
wait
Turn of another player:
      YOUR AREA
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ * * * ~ ~ ~
1 ~ ~ ~ ~ ~ ~ * X * ~ o
2 ~ ~ o ~ o * * * ~ ~ ~
3 ~ ~ ~ ~ o ~ ~ ~ ~ ~ o
4 o ~ ~ ~ ~ ~ ~ ~ ~ ~ o
5 ~ ~ ~ ~ ~ ~ X o ~ ~ o
6 ~ ~ o ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ o ~ ~ ~ ~ ~ ~ ~ ~
8 o ~ o ~ ~ ~ o o o o o
9 o ~ ~ ~ * ~ ~ ~ ~ ~ ~
Wait for your turn
your_turn
      YOUR AREA
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ * * * ~ ~
1 ~ ~ ~ ~ ~ ~ * X * ~ o
2 ~ ~ o ~ o * * * ~ ~ ~
3 ~ ~ ~ ~ o ~ ~ ~ ~ ~ o
4 o ~ ~ ~ ~ ~ ~ ~ ~ ~ o
5 ~ ~ ~ ~ ~ ~ X o ~ ~ o
6 ~ ~ o ~ ~ * ~ ~ ~ ~ ~
7 ~ ~ o ~ ~ ~ ~ ~ ~ ~ ~
8 o ~ o ~ ~ ~ o o o o o
9 o ~ ~ ~ * ~ ~ ~ ~ ~ ~
Your turn:
      ENEMY
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ * ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
9 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

Enter your command:
>shot 1 1
Wrong command!
Wrong coordinates!

Enter your command:
>shoot 1 1
miss
      ENEMY
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
1 ~ * ~ ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ * ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
```

```
9 ~ ~ ~ ~ * ~ ~ ~ ~ ~
Turn of another player:
Wait for your turn
your_turn
      YOUR AREA
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ o ~ ~ o ~
1 ~ * ~ ~ ~ ~ ~ ~ ~ ~
2 o ~ o ~ ~ ~ ~ ~ ~ ~
3 o ~ o ~ ~ o o ~ ~ ~
4 o ~ o ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ o o o o
6 ~ o * o ~ ~ ~ ~ ~ ~
7 ~ ~ ~ o ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ o o ~
9 ~ o ~ ~ ~ ~ ~ ~ ~ ~
Your turn:
      ENEMY
  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ * * * ~ ~
1 ~ ~ ~ ~ ~ * X * ~ ~
2 ~ ~ ~ ~ ~ * * * ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ X ~ ~ ~ ~
6 ~ ~ ~ ~ ~ * ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
9 ~ ~ ~ ~ * ~ ~ ~ ~ ~

Enter your command:
>
```

```
8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
9 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
Turn of another player:
Wait for your turn
```

Выводы:

При выполнении данного курсового проекта для проектирования клиент-серверной игры было удобно использовать сервер очередей ZeroMQ. Он позволяет независимо обрабатывать сообщения от разных клиентов и предоставляет удобную абстракцию над сокетами, благодаря чему есть возможность легко реализовать обмен сообщениями между сервером и клиентами. При этом сервер может поддерживать несколько игр одновременно.

Основным недостатком использования сервера сообщений оказалась невозможность игнорировать ввод команды во время хода другого игрока, и, таким образом, введенная команда обработывается сразу же при возвращении хода первому игроку.