

Московский авиационный институт
(национальный исследовательский университет)

Институт: «Информационные технологии и прикладная
математика»

Кафедра: 806 «Вычислительная математика и
программирование»

Курсовой проект
по курсу «Операционные системы»

Игра «Змейка» для двух игроков

Студент: Лагуткина Мария Сергеевна

Группа: М8О-206Б-19

Преподаватель: Соколов А. А.

Дата:

Оценка:

Москва, 2020

1 Постановка задачи

Цель курсового проекта

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

Задание

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

Выбранная тема: интерактивная игра «Змейка» для двух игроков, где взаимодействие между игроками происходит с помощью сервера сообщений (с помощью библиотеки обмена сообщениями ZeroMQ).

2 Общий метод и алгоритм решения

Программа запускается для каждого игрока: один (как клиент) подключается к другому (как сервер), это выбирается при запуске программы: 1 - подключиться, 2 - создать игру. Создается сокет, и передается от клиента к серверу данное значение - как сигнал подключения клиента. Игрок, который подключается к другому, должен ввести IP игрока-сервера.

Затем каждая сторона вводит username, клиент и сервер обмениваются этими значениями, для корректности отображения счета в дальнейшем.

Далее отрисовывается поле, яблоко, змейки и счет (поле отрисовывается один раз, далее перерисовываются только изменяющиеся части). Змейка игрока - зеленая, змейка противника - фиолетовая. Змейки представляют собой связный список, каждый узел которого содержит координаты узла и указатель на следующий узел. При нажатии клавиш - изменяется направление змейки, после чего происходит обмен координатами головы между сервером и клиентом. При съедании яблока - оно генерируется в новом месте и увеличивается счет у игрока, чья змейка съела яблоко. Игра идет, пока не будут съедены все яблоки (их текущее количество отображается под счетом игроков). Для отрисовки графики была использован заголовочный файл `conio.h`. Для реализации взаимодействия сервер-клиент используются следующие системные вызовы:

1. `zmq_ctx_new ()` - создает новый контекст
2. `zmq_setsockopt` - устанавливает флаги для сокета
3. `zmq_bind ()` создает конечную точку для принятия соединений и привязывает ее к сокету

4. `zmq_msg_recv` - получить сообщение из сокета
5. `zmq_msg_send` - отправить сообщение
6. `zmq_msg_close()` информирует инфраструктуру `zmq` о том, что любые ресурсы, связанные с объектом сообщения, на который ссылается `msg`, больше не требуются и могут быть освобождены

3 Основные файлы программы

`snake.cpp`

```
1 // snake.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается
  // выполнение программы.
2
3 //змейка для двух пользователей: первый клиент становится сервером, второй
  //подключается к первому.
4 //игра заканчивается, как только будут съедены все 25 яблок, побеждает тот, у кого
  //счет больше.
5
6 #define _CRT_SECURE_NO_WARNINGS
7 #include <iostream>
8 #include "zmq.h"
9 #include <assert.h>
10 #include <Windows.h>
11 #include <conio.h>
12 #include <string>
13 #include <cstring>
14
15 int height = 25;
16 int width = 100;
17
18 int gameover = 0, counter, gameover2 = 0, counter2, conn;
19 int status = 0;
20 int lflag = 0, rflag = 0, uflag = 0, dflag = 0;
21 int lflag2 = 0, rflag2 = 0, uflag2 = 0, dflag2 = 0;
22 short fcount;
23 void *context;
24 void *serverSocket;
25
26 using namespace std;
27
28 class Snake {
29     int x, y, fx, fy, x2, y2;
30     char playername[50], playername2[50];
31
32     struct node {
33         int nx, ny;
```

```

34     struct node *next;
35     struct node *back;
36 };
37 struct node *head = NULL;
38 struct node *head2 = NULL;
39
40 void gotoxy(int x, int y) {
41     COORD pos = { x,y };
42     SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), pos);
43 }
44
45 void nameandscore() {
46     textcolour(6);
47     gotoxy(101, 2);
48     cout << playername << "'s SCORE = " << counter * 100;
49     gotoxy(101, 4);
50     cout << playername2 << "'s SCORE = " << counter2 * 100;
51     gotoxy(101, 6);
52     cout << "Remained Fruit :";
53     gotoxy(117, 6);
54     cout << " ";
55     gotoxy(117, 6);
56     cout << fcount;
57 }
58
59 void textcolour(int k) {
60     HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
61     SetConsoleTextAttribute(hConsole, k);
62 }
63
64 public:
65     void window() {
66         textcolour(1);
67         for (int i = 0; i <= width; i++){
68             gotoxy(i, 0);
69             cout << "#";
70             gotoxy(i, height);
71             cout << "#";
72         }
73         for (int i = 0; i <= height; i++){
74             gotoxy(0, i);
75             cout << "#";
76             gotoxy(width, i);
77             cout << "#";
78         }
79     }
80
81     void setup(){ //begin
82         resetflag2();

```

```

83     gameover2 = 0;
84     counter2 = 0;
85     fcount = 25;
86     head2 = new node;
87     head2->nx = width / 2 + 5;
88     head2->ny = height / 2 + 5;
89     head2->next = NULL;
90     head2->back = NULL;
91     x2 = width / 2 + 5;
92     y2 = height / 2 + 5;
93     counter = 0;
94     gameover = 0;
95     window();
96     resetflag();
97     nameandscore();
98     head = new node;
99     head->nx = width / 2;
100    head->ny = height / 2;
101    head->next = NULL;
102    head->back = NULL;
103    x = width / 2;
104    y = height / 2;
105    label1:
106        fx = rand() % width;
107        if (fx == 0 || fx == width)
108            goto label1;
109    label2:
110        fy = rand() % height;
111        if (fy == 0 || fy == height)
112            goto label2;
113    }
114
115    void drawlist(struct node *h, int k){ //drawing snake
116        textcolour(k);
117        struct node *ptr;
118        ptr = h;
119        while (ptr != NULL){
120            gotoxy(ptr->nx, ptr->ny);
121            cout << "o";
122            ptr = ptr->next;
123        }
124    }
125
126    void destroylist(struct node *h) {
127        struct node *ptr;
128        ptr = h;
129        while (ptr != NULL){
130            gotoxy(ptr->nx, ptr->ny);
131            cout << " ";

```

```

132         ptr = ptr->next;
133     }
134 }
135
136 void draw(){
137     drawlist(head, 2);
138     drawlist(head2, 5);
139     gotoxy(fx, fy);
140     textcolour(4);
141     cout << "*";
142     Sleep(70);
143     destroylist(head);
144     destroylist(head2);
145 }
146
147 void resetflag(){
148     uflag = 0;
149     dflag = 0;
150     lflag = 0;
151     rflag = 0;
152 }
153
154 void resetflag2(){
155     uflag2 = 0;
156     dflag2 = 0;
157     lflag2 = 0;
158     rflag2 = 0;
159 }
160
161 void play(){
162     int h;
163     char ch;
164     if (_kbhit()){
165         ch = _getch();
166         h = ch;
167         switch (h){
168             case 72:if (dflag != 1) { resetflag(); uflag = 1; }
169                     break;
170             case 80:if (uflag != 1) { resetflag(); dflag = 1; }
171                     break;
172             case 75:if (rflag != 1) { resetflag(); lflag = 1; }
173                     break;
174             case 77:if (lflag != 1) { resetflag(); rflag = 1; }
175                     break;
176             default:break;
177         }
178     }
179 }
180

```

```

181 void connection1() {
182     textcolour(6);
183     gotoxy(width / 2 - 18, height / 2 - 3);
184     cout << "WELCOME TO SNAKE GAME MULTIPLAYER MODE ";
185     textcolour(8);
186     gotoxy(width / 2 - 18, height / 2);
187     cout << "You can you can connect to another player or start your own game";
188     gotoxy(width / 2 - 18, height / 2 + 2);
189     cout << "Enter connect or create new game (1 or 2): ";
190     cin >> conn;
191     string ip;
192     if (conn == 1) { //connect
193         gotoxy(width / 2 - 13, height / 2 + 4);
194         cout << "Enter player's IP: ";
195         cin >> ip;
196
197         context = zmq_ctx_new();
198         serverSocket = zmq_socket(context, ZMQ_REQ);
199         string addr = "tcp://" + ip + ":4040";
200         int rc = zmq_connect(serverSocket, addr.c_str());
201
202         // WAITING PLAYERS
203         status = 1;
204         zmq_msg_t zmqMessage;
205         zmq_msg_init_size(&zmqMessage, sizeof(int));
206         memcpy(zmq_msg_data(&zmqMessage), &status, sizeof(int));
207         int send = zmq_msg_send(&zmqMessage, serverSocket, 0);
208         zmq_msg_close(&zmqMessage);
209
210         zmq_msg_t reply;
211         zmq_msg_init(&reply);
212         zmq_msg_recv(&reply, serverSocket, 0);
213         zmq_msg_close(&reply);
214
215     }
216     else if (conn == 2) { //create game
217         int Timeout = 2000;
218         context = zmq_ctx_new();
219         serverSocket = zmq_socket(context, ZMQ_REP);
220
221         zmq_setsockopt(serverSocket, ZMQ_LINGER, &Timeout, sizeof(Timeout));
222         zmq_bind(serverSocket, "tcp://*:4040");
223
224         // WAITING PLAYERS
225         gotoxy(width / 2 - 13, height / 2 + 4);
226         cout << "Wait....";
227
228         zmq_msg_t stat;
229         zmq_msg_init(&stat);

```

```

230         zmq_msg_recv(&stat, serverSocket, 0);
231         int *m = (int *)zmq_msg_data(&stat);
232         zmq_msg_close(&stat);
233
234         zmq_msg_t reply;
235         zmq_msg_init_size(&reply, strlen("ok") + 1);
236         memcpy(zmq_msg_data(&reply), "ok\0", 3);
237         zmq_msg_send(&reply, serverSocket, 0);
238         zmq_msg_close(&reply);
239     }
240     else { cout << "Enter correct number"; }
241     system("cls");
242 }
243
244 void welcome(){
245     connection1();
246     textcolour(8);
247     gotoxy(width / 2 - 15, height / 2);
248     cout << "Enter Player Name : ";
249     gotoxy(width / 2 + 7, height / 2);
250     cin >> playername;
251
252     char* ptr_playername;
253
254     if (status == 1) { //client
255         zmq_msg_t zmqMessage;
256         zmq_msg_init_size(&zmqMessage, sizeof(playername));
257         memcpy(zmq_msg_data(&zmqMessage), &playername, sizeof(playername));
258         int send = zmq_msg_send(&zmqMessage, serverSocket, 0);
259         zmq_msg_close(&zmqMessage);
260
261         zmq_msg_t reply;
262         zmq_msg_init(&reply);
263         zmq_msg_recv(&reply, serverSocket, 0);
264         ptr_playername = (char *)zmq_msg_data(&reply);
265         zmq_msg_close(&reply);
266     }
267     else {
268         zmq_msg_t stat;
269         zmq_msg_init(&stat);
270         zmq_msg_recv(&stat, serverSocket, 0);
271         ptr_playername = (char *)zmq_msg_data(&stat);
272
273         zmq_msg_close(&stat);
274
275         zmq_msg_t reply;
276         zmq_msg_init_size(&reply, sizeof(playername));
277         memcpy(zmq_msg_data(&reply), &playername, sizeof(playername));
278         zmq_msg_send(&reply, serverSocket, 0);

```



```

279         zmq_msg_close(&reply);
280     }
281     sizeof(ptr_playername));
282     strcpy(playername2, ptr_playername);
283     system("cls");
284 }
285
286 char end(){
287     char c;
288     gotoxy(width / 2 - 5, height / 2 - 4);
289     cout << "GAME OVER \n";
290     textcolour(1);
291     gotoxy(width / 2 - 15, height / 2 - 2);
292     cout << playername << " You Scored : " << counter * 100;
293     gotoxy(width / 2 - 15, height / 2);
294     cout << playername2 << " You Scored : " << counter2 * 100;
295     textcolour(4);
296     if (gameover != 0) {
297         gotoxy(width / 2 - 15, height / 2 + 2);
298         cout << playername << " has lost !";
299     }
300     else{
301         gotoxy(width / 2 - 15, height / 2 + 2);
302         cout << playername2 << " has lost !";
303     }
304     if (fcount == 0){
305         textcolour(4);
306         gotoxy(width / 2 - 15, height / 2 + 2);
307         if (counter > counter2){
308             cout << playername << " has WON !";
309         }
310         else{
311             cout << playername2 << " has WON !";
312         }
313     }
314     textcolour(6);
315     gotoxy(width / 2 - 15, height / 2 + 4);
316     cout << "Want To Play Again ? (Y/N) : ";
317     cin >> c;
318     system("cls");
319     return c;
320 }
321
322 struct xy {
323     int x;
324     int y;
325 };
326
327 xy coords_node1;

```

```

328     xy coords_node2;
329     xy *c_node2;
330
331     void run(){
332         if (uflag == 1)
333             y--;
334         else if (dflag == 1)
335             y++;
336         else if (lflag == 1)
337             x--;
338         else if (rflag == 1)
339             x++;
340
341         coords_node1.x = x;
342         coords_node1.y = y;
343
344         if (status == 1) { //client
345             zmq_msg_t zmqMessage;
346             zmq_msg_init_size(&zmqMessage, sizeof(xy));
347             memcpy(zmq_msg_data(&zmqMessage), &coords_node1, sizeof(xy));
348             int send = zmq_msg_send(&zmqMessage, serverSocket, 0);
349             zmq_msg_close(&zmqMessage);
350
351             zmq_msg_t reply;
352             zmq_msg_init(&reply);
353             zmq_msg_recv(&reply, serverSocket, 0);
354             c_node2 = (xy *)zmq_msg_data(&reply);
355             zmq_msg_close(&reply);
356         }
357         else {
358             zmq_msg_t stat;
359             zmq_msg_init(&stat);
360             zmq_msg_recv(&stat, serverSocket, 0);
361             c_node2 = (xy *)zmq_msg_data(&stat);
362
363             zmq_msg_close(&stat);
364
365             zmq_msg_t reply;
366             zmq_msg_init_size(&reply, sizeof(xy));
367             memcpy(zmq_msg_data(&reply), &coords_node1, sizeof(xy));
368             zmq_msg_send(&reply, serverSocket, 0);
369             zmq_msg_close(&reply);
370
371         }
372         x2 = c_node2->x;
373         y2 = c_node2->y;
374     }
375
376     void dolist(struct node *h, int pp, int qq) {

```

```

377     struct node *ptr, *prev;
378     ptr = h;
379     prev = h;
380
381     while (ptr->next != NULL) {
382         prev = ptr;
383         ptr = ptr->next;
384     }
385     while (prev != h){
386         ptr->nx = prev->nx;
387         ptr->ny = prev->ny;
388         prev = prev->back;
389         ptr = ptr->back;
390     }
391     ptr->nx = prev->nx;
392     ptr->ny = prev->ny;
393     prev->nx = pp;
394     prev->ny = qq;
395 }
396
397 void drawagain() {
398     if (x == width) {
399         x = 1;
400     }
401     if (x == 0) {
402         x = width - 1;
403     }
404     if (y == 0) {
405         y = height - 1;
406     }
407     if (y == height) {
408         y = 1;
409     }
410 }
411
412 void drawagain2() {
413     if (x2 == width) {
414         x2 = 1;
415     }
416     if (x2 == 0) {
417         x2 = width - 1;
418     }
419     if (y2 == 0) {
420         y2 = height - 1;
421     }
422     if (y2 == height) {
423         y2 = 1;
424     }
425 }

```

```

426
427 void generatefruit() {
428     label1:
429         fx = rand() % width;
430         if (fx == 0)
431             goto label1;
432     label2:
433         fy = rand() % height;
434         if (fy == 0)
435             goto label2;
436 }
437
438 void checkfcount() {
439     if (fcount == 0) {
440         gameover = 1;
441         gameover2 = 1;
442     }
443 }
444
445 void checkup() {
446     drawagain();
447     if (x == fx && y == fy){
448         fcount--;
449         checkfcount();
450         struct node *t, *ptr, *prev;
451         t = new node;
452         t->next = NULL;
453         t->back = NULL;
454         ptr = head;
455         prev = head;
456         while (ptr->next != NULL){
457             ptr = ptr->next;
458         }
459         ptr->next = t;
460         t->back = ptr;
461         generatefruit();
462         counter++;
463         nameandscore();
464     }
465     dolist(head, x, y);
466 }
467
468 void checkup2(){
469     drawagain2();
470     if (x2 == fx && y2 == fy){
471         fcount--;
472         checkfcount();
473         struct node *t, *ptr, *prev;
474         t = new node;

```

```

475         t->next = NULL;
476         t->back = NULL;
477         ptr = head2;
478         prev = head2;
479         while (ptr->next != NULL) {
480             ptr = ptr->next;
481         }
482         ptr->next = t;
483         t->back = ptr;
484         generatefruit();
485         counter2++;
486         nameandscore();
487     }
488     dolist(head2, x2, y2);
489 }
490
491 void game(){
492     char ch;
493     welcome();
494     do {
495         setup();
496         window();
497         while (gameover != 1 && gameover2 != 1){
498             draw();
499             play();
500             run();
501             checkup();
502             checkup2();
503         }ch = end();
504     } while (ch == 'y' || ch == 'Y');
505 }
506
507 };
508 int main(){
509
510     Snake s;
511     s.game();
512     system("exit");
513 }

```

4 Вывод

При работе над курсовой я чуть больше погрузилась в реализацию взаимодействия клиент-сервер. ZeroMQ - удобный инструмент для организации такого взаимодействия. Так же мною были изучены принципы написания интерактивной игры. Хотя и змейка - достаточно примитивная игра, для ее написания требуется продумать

цикл игры. Для красивого отображения игрового поля был использован заголовочный файл `copio.h`, с которым я ранее не работала, но который позволяет достаточно просто задавать цвета и положение курсора при изменении параметров в игре.