

(1) 网络游戏能够正常运行和演示 (60%)；

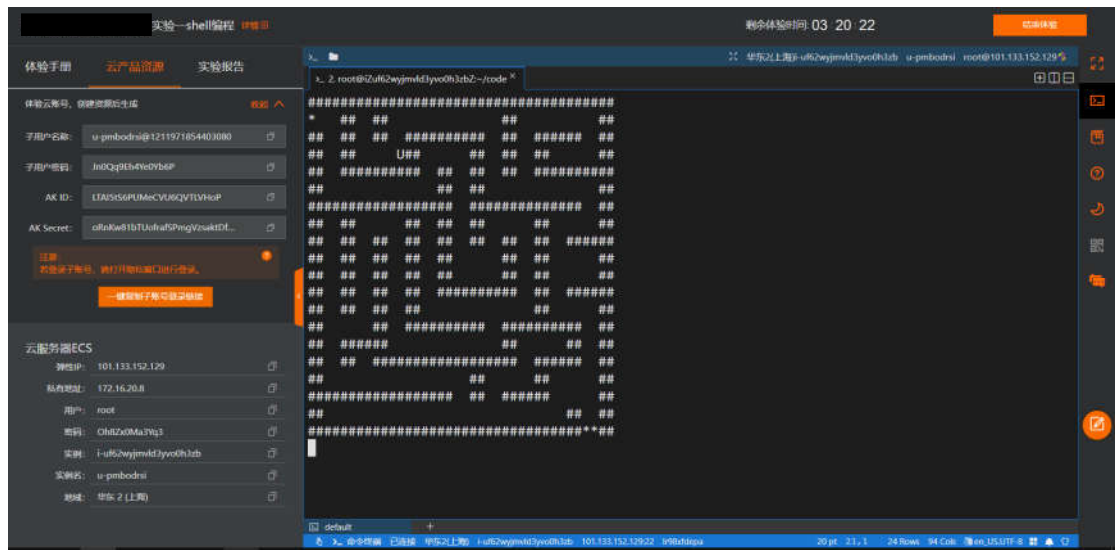


图 1 在线网页展示

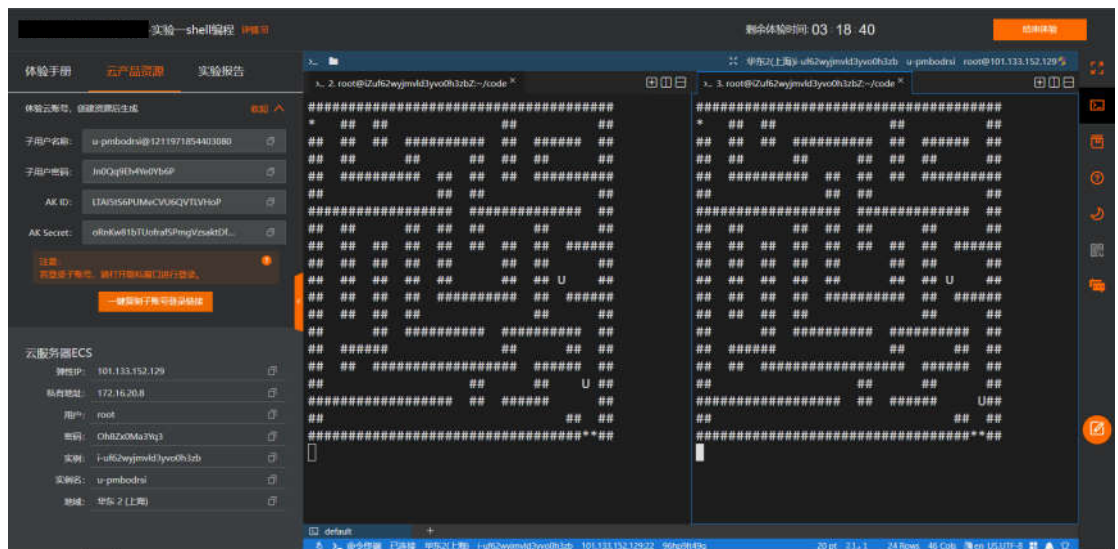


图 2 多用户展示 1

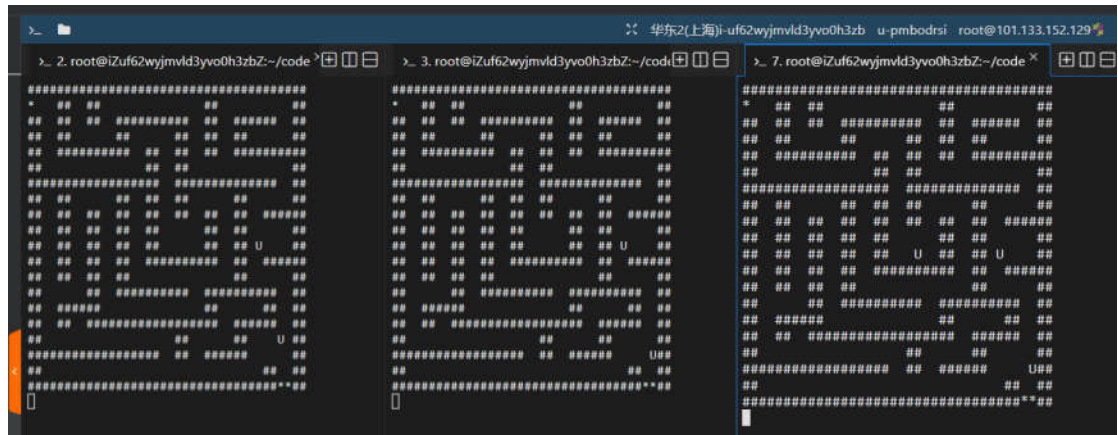


图 3 多用户展示 2

### 游戏说明:

迷宫难题是一个多用户的游戏，可以让很多人在线同时走迷宫，并且当所有人都到达终点时游戏结束。

玩家可以通过 wasd 或者方向键上下左右来控制自己的角色，在游戏中，每个角色都用“U”来显示。

(2) 程序说明:

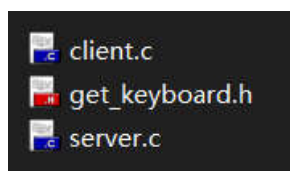


图 4 源代码

### 运行过程:

首先运行 `server.c`, `cc server.c -o server`

再运行 server 8888

接着运行客户端 `client.c`, `cc client.c -o client`

## 运行 client

这样子就可以进入游戏界面了，其他用户也是一样。

代码说明:

[illegible]

图 5 数据结构

游戏的所有数据都保存在服务器中，每次玩家移动服务器都会向玩家返回地图数据。

```

void move(int id,int nrow, int ncol, int nnrow, int nncol) {
    if(0==boards[nrow][ncol] || 3==boards[nrow][ncol]) { //0 1
        boards[nrow][ncol] +=7;// in
        boards[user[id].row][user[id].col] -=7;//out
        user[id].row = nrow;
        user[id].col = ncol;
    }
    int i;
    for(i=0; i<H; i++) { //judge
        int j;
        for(j=0; j<L; j++) {
            if(10==boards[i][j]||-4==boards[i][j]) {
                cnt++;
            }
            if(userSum()==cnt) {
                printf("I'm 10£i\n");
                close(user[id].fd);
            }
        }
    }
    cnt=0; //judge again
}
}

```

图 6 玩家移动

```

int dir=atoi(buf);
int id=getId(infd);
switch(dir) { //judge position
    case KEY_W:
    case KEY_w:
    case KEY_UP:
        move(id,user[id].row-1, user[id].col, user[id].row-2, user[id].col);
        break;
    case KEY_S:
    case KEY_s:
    case KEY_DOWN:
        move(id,user[id].row+1, user[id].col, user[id].row+2, user[id].col);
        break;
    case KEY_A:
    case KEY_a:
    case KEY_LEFT:
        move(id,user[id].row, user[id].col-1, user[id].row, user[id].col-2);
        break;
    case KEY_D:
    case KEY_d:
    case KEY_RIGHT:
        move(id,user[id].row, user[id].col+1, user[id].row, user[id].col+2);
        break;
    case KEY_q:
        exit(0);//exit
}
write(infd, boards, sizeof(boards));

```

图 7 服务器的核心——游戏逻辑的实现

```

//print
void print(int boards[][L]) {
    int i,j;
    for(i=0; i<H; i++) {
        for(j=0; j<L; j++) {
            switch(boards[i][j]) {
                case 0:
                    printf(" ");
                    break; //rode
                case 1:
                    printf("#");
                    break; //wall
                case 3:
                    printf("*");
                    break; //desc
                case 10:
                case 7:
                    printf("U");
                    break; //user
            }
            printf("%d",boards[i][j]);
        }
        printf("\n");
    }
}

```

图 8 客户端的主要任务

```

while(1) {
    memset(buf,0,sizeof(buf));
    if(num>0){
        read (sockFd, boards, sizeof(boards));
        print(boards);
        num--;
    }
    int dir=get_keyboard();
    system("clear");
    sprintf(buf,"%d%c",dir,'\0');
    ret = write(sockFd,buf,sizeof(buf));
    if(ret < 0) {
        perror("write error!");
        close(sockFd);
        return -1;
    }
    read (sockFd, boards, sizeof(boards));
    print(boards);
}

```

图 9 客户端的核心——收发数据

(3) 整个项目的完成度及从用户角度看是否好玩（10%）；

这个项目具有很高的完成度，实现了玩家加入迷宫、上下左右移动、限制移动、游戏结束退出等功能。



从用户角度来说，易上手，没有复杂的操作，能够流畅的操作，不卡顿。

(4) 应用层协议设计 (5%)；

游戏在传输数据时，采用的是多种数据格式，分别为，服务器向客户端发送地图数据、客户端向服务器发送操作指令等。

服务器发送的数据具体为一个二维数组，包含了地图的所有信息。

客户端发送的数据是一个一个操作指令，包含了常用的来控制方向的按键。

(5) 采用技术的难度 (5%)；

服务器采用的是 epoll，实现了多路复用的 I/O 技术。

客户端采用的是无回显键盘输入。

(6) 性能、健壮性和 bug 情况 (10%)；

性能：

理论上，可以加入的用户没有上限，但是当地图没有空位来生成用户时，达到上限。

健壮性：

目前还没有出现过玩家冲出围墙的情况，但是当部分玩家到达终点时，这一部分玩家还可以来回移动。

BUG：

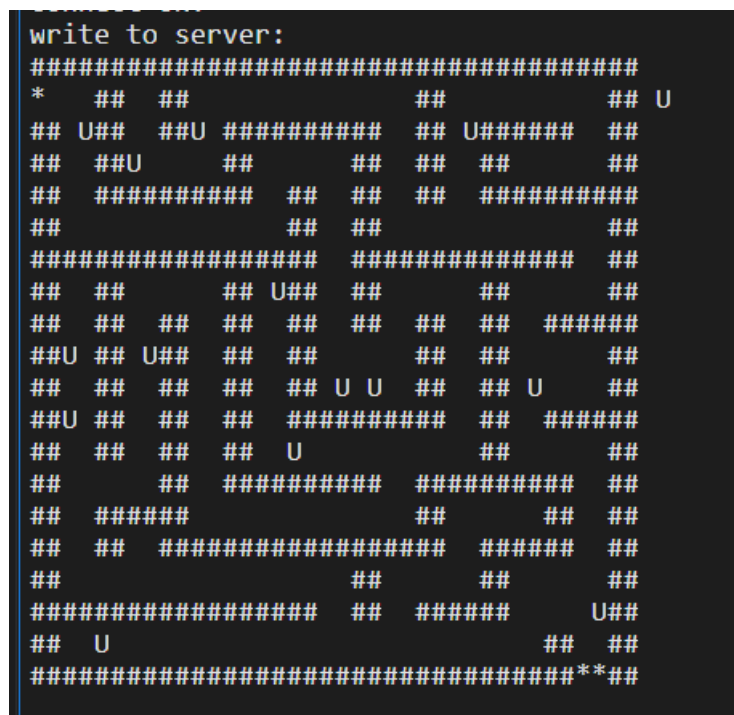


图 10 Bug

这是因为二维数组没有完全利用起来，并且玩家是随机刷新的，所有有可能会刷新到没有使用的区域。解决办法，限制数组的大小。