УДК 004.42

Разработка тактической клиент-серверной игры на Android

Семченко Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Научный руководитель
Лучанинов Дмитрий Васильевич
Приамурский государственный университет имени Шолом-Алейхема
Старший преподаватель кафедры информационных систем, математики и правовой информатики

Аннотация
В данной статье описан метод написания клиент серверной игры для андроид смартфона в среде разработке Android studio на языке программирования java script. Практическим результатом является рабочее мобильное приложение с возможностью поиграть с помощью или без интернета в карточную игру.
Ключевые слова: Андроид, игра, клиент, сервер

Android tactical client-server game development

Semchenko Regina Viktorovna
Sholom-Aleichem Priamursky State University
student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Scientific adviser
Luchaninov Dmitry Vasilyevich
Sholom-Aleichem Priamursky State University
Senior lecturer of the Department of Information Systems, Mathematics and Law Informatics

Abstract
This article describes a method for writing a client server game for android smartphone in the development environment Android studio in the programming

language javascript. The bottom line is a working mobile application with the opportunity to play a card game with or without the Internet.
Keywords: Android, game, client, server

На сегодняшний день стали активно развиваться облачные технологии. Большое программное обеспечение становится более доступным, благодаря множественному количеству сетевых платформ и сервисов. Это означает, что не обязательно устанавливать на компьютер или смартфон полноценную версию продукта, а для доступа к ним, необходим всего лишь интернет.

Сам сервер – это компьютер, который берет на себя выполнение специализированных функций, он прослушивает порты на подключение. Порт – это тока доступа к серверу, к которой подключаются клиенты.

Цель исследования – создание клиент серверной игры для андроид смартфона в AndroidStudio.

Объект исследования – клиент серверная игра.

Предмет исследования – создание клиент серверной игры для андроид смартфона в AndroidStudio.

Задачи исследования:

- Изучить научно-техническую литературу.
- Разработать алгоритм клиент-серверной игры.
- Создать клиент-серверную игру.

Методы исследования:

- Изучение научно-технической литературы.
- Разработка алгоритма клиент-серверной игры на языке программирования JavaScript.

Практическая значимость разработки алгоритма «клиент-сервер» состоит в его применении при разработке многих онлайн игр. Данный алгоритм будет иметь хорошие шансы на качественную реализацию проекта по разработке игры.

В своей статье Е.В.Непомнящих привел анализ требований, которые предъявляются к современным пошаговым онлайн играм. А так же рассмотрел наиболее разумные пути их разработки. [1]. К.В. Эммануилович и В.Ю. Плотников разработали программу предназначенную для тестирования эвристических алгоритмов поиска на больших графах и для интеллектуальной игры. Клиент-серверная архитектура программы «легких клиент» позволяла запускать клиент на всех браузерах [2]. В статье Перлмана Стефена и Ван Дер Роджера была разработана программа позволяющая обеспечить потоковое интерактивное видео, результат заключается в повышении плавности воспроизведения потокового интерактивного видео на устройстве. [3]. И.А.Пушкин описал разработку серверной части многопользовательской игры, провел полный разбор кода в ходе проектирования игрового сервера. [4]. Clypool Mark провел анализ сети, на которую влияет большое количество людей подключенных к одному серверу, анализ показал, что большинство игровых серверов имеют задержки, таких как стратегии в реальном времени, спортивные и ролевые

игры, предоставляли многочисленные варианты выбора серверов для игроков [5]. Webb Steven Daniel и Soh Sieteng реализовали архетектуру зеркального сервера для сетевых игр. Им удалось найти оптимально сопоставление зеркальных серверов и увеличить пропускную способность и уменьшить задержку между клиентом и сервером [6].

Теперь напишем код, который будет образовывать сервер на телефоне. Сервер-клиент был написан с использованием сокетов Java. На одном устройстве работает сервер (и клиент), а на других - клиенты. Данные передаются в виде реализованных объектов. Эти объекты имеют всю информацию, необходимую для воссоздания экземпляра игры (рис.1).

```
package srk.syracuse.gameofcards.Connections;


import ...


public class ServerConnectionThread extends Thread {

    static final int SocketServerPORT = 8080;
    public static HashMap<Socket, String> socketUserMap = new HashMap();
    public static boolean serverStarted = false;
    public static ServerSocket serverSocket;
    public static boolean allPlayersJoined = false;

    public ServerConnectionThread() {

    }

    @Override
    public void run() {
        if (serverSocket == null) {
            try {
                serverSocket = new ServerSocket(SocketServerPORT);
                serverStarted = true;
                while (true) {
                    Socket socket = serverSocket.accept();
                    if (!allPlayersJoined) {
                        Thread socketListenThread = new Thread(new ServerListenerThread(socket));
                        socketListenThread.start();
                        ServerSenderThread sendGameName = new ServerSenderThread(socket, HostFragment.gameName.getText().toString());
                        sendGameName.start();
                        socketUserMap.put(socket, null);
                        if (socketUserMap.size() == HostFragment.numberPlayers) {
                            allPlayersJoined = true;
                        }
                    }
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Рис. 1 – Соединительная часть сервера

Сервер поддерживает один поток слушателя и отправителя для каждого клиента, так как сокеты для каждого клиента различны. Код, представленный на рисунке 4, заботится о прослушивании и отправке. Точно так же у нас есть два потока, работающих на каждом клиенте с той же целью (рис.2).

```java
package srk.syracuse.gameofcards.Connections;

import ...

public class ServerListenerThread extends Thread {

    private Socket hostThreadSocket;

    ServerListenerThread(Socket soc) { hostThreadSocket = soc; }

    @Override
    public void run() {
        while (true) {
            ObjectInputStream objectInputStream;
            try {
                InputStream inputStream = null;
                inputStream = hostThreadSocket.getInputStream();
                objectInputStream = new ObjectInputStream(inputStream);
                Object gameObject;
                Bundle data = new Bundle();
                gameObject = objectInputStream.readObject();
                if (gameObject != null) {
                    if (gameObject instanceof PlayerInfo) {
                        data.putSerializable(Constants.DATA_KEY, (PlayerInfo) gameObject);
                        data.putInt(Constants.ACTION_KEY, Constants.PLAYER_LIST_UPDATE);
                        ServerConnectionThread.socketUserMap.put(hostThreadSocket, ((PlayerInfo) gameObject).username);
                    } else {
                        data.putSerializable(Constants.DATA_KEY, (Game) gameObject);
                    }
                    Message msg = new Message();
                    msg.setData(data);
                    HostFragment.serverHandler.sendMessage(msg);
                }

            } catch (IOException e) {
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Рис. 2 – Прослушивание данных

Сервер получает GameObject и распространяет его среди всех клиентов. Когда клиенты получают этот GameObject, они просто обновляют свои состояния (рис.3).

```
package srk.syracuse.gameofcards.Connections;

import ...

public class ServerSenderThread extends Thread {

    private Socket hostThreadSocket;
    Object message;

    public ServerSenderThread(Socket socket, Object message) {
        hostThreadSocket = socket;
        this.message = message;
    }

    @Override
    public void run() {
        OutputStream outputStream;
        ObjectOutputStream objectOutputStream;

        try {
            outputStream = hostThreadSocket.getOutputStream();
            objectOutputStream = new ObjectOutputStream(outputStream);
            objectOutputStream.writeObject(message);
            if (message instanceof Game) {
                PlayerListFragment.gameObject = (Game) message;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }


    }

}
```

Рис. 3 – Отправка игровых данных

Далее осталось написать так же соединительную часть, прослушку данных и отправку данных с клиентской части (рис.4-6).

```
package srk.syracuse.gameofcards.Connections;


import ...

public class ClientConnectionThread extends Thread {
    public static Socket socket;
    String dstAddress;
    int dstPort = 8080;
    public static boolean serverStarted = false;
    String userName;

    public ClientConnectionThread(String userName) { this.userName = userName; }

    @Override
    public void run() {
        if (socket == null) {
            try {
                ArrayList<String> deviceList = WifiHelper.getDeviceList();
                if (deviceList.size() > 0) {
                    dstAddress = deviceList.get(0);
                    if (dstAddress != null) {
                        socket = new Socket(dstAddress, dstPort);
                        if (socket.isConnected()) {
                            serverStarted = true;
                            ClientListenerThread clientListener = new ClientListenerThread(socket);
                            clientListener.start();
                            PlayerInfo playerInfo = new PlayerInfo(userName);
                            ClientSenderThread sendUserName = new ClientSenderThread(socket, playerInfo);
                            sendUserName.start();
                        }
                    }
                }
            } catch (UnknownHostException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

}
```

Рисунок 4 – Соединительная часть клиента

```java
package srk.syracuse.gameofcards.Connections;


import ...

public class ClientListenerThread extends Thread {

    Socket socket;

    ClientListenerThread(Socket soc) { socket = soc; }

    @Override
    public void run() {
        try {
            while (true) {
                ObjectInputStream objectInputStream;
                InputStream inputStream = null;
                inputStream = socket.getInputStream();
                objectInputStream = new ObjectInputStream(inputStream);
                Bundle data = new Bundle();
                Object serverObject = (Object) objectInputStream.readObject();
                if (serverObject != null) {
                    if (serverObject instanceof String) {
                        data.putSerializable(Constants.DATA_KEY, (String) serverObject);
                        data.putInt(Constants.ACTION_KEY, Constants.UPDATE_GAME_NAME);
                    }
                    if (serverObject instanceof Game) {
                        data.putSerializable(Constants.DATA_KEY, (Game) serverObject);
                    }
                    Message msg = new Message();
                    msg.setData(data);
                    MainFragment.clientHandler.sendMessage(msg);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

Рис. 5 – Прослушивание данных с клиента

```java
package srk.syracuse.gameofcards.Connections;

import ...

public class ClientSenderThread extends Thread {

    private Socket hostThreadSocket;
    Object message;
    public static boolean isActive = true;

    public ClientSenderThread(Socket socket, Object message) {
        hostThreadSocket = socket;
        this.message = message;
    }

    @Override
    public void run() {
        OutputStream outputStream;
        ObjectOutputStream objectOutputStream;
        if (hostThreadSocket.isConnected()) {
            try {
                if (isActive) {
                    if (message instanceof Game && !Constants.isPlayerActive(MainFragment.userName.getText().toString(), (Game) message)) {
                        isActive = false;
                    }
                    outputStream = hostThreadSocket.getOutputStream();
                    objectOutputStream = new ObjectOutputStream(outputStream);
                    objectOutputStream.writeObject(message);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }

        }

    }


}
```

Рисунок 6 – отправка данных от клиента

После того как были написаны клиент и сервер, осталось добавить картинки как будет выглядеть игра и ее части.

После добавление оставшегося материала, игра готова (рис.7-9).

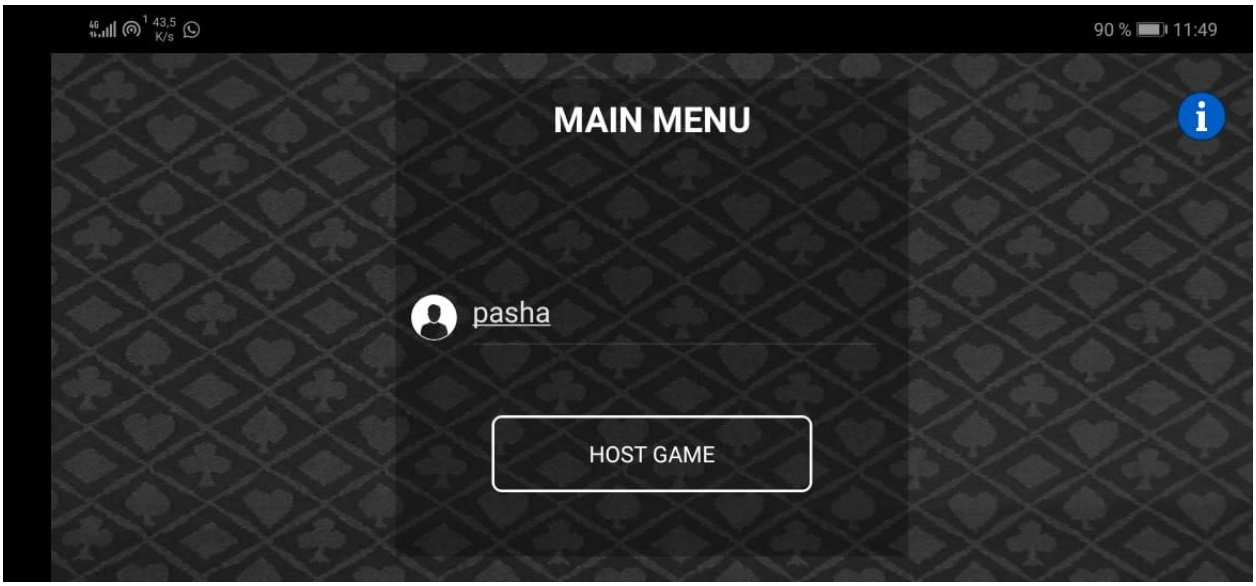

Рисунок 7 – Игра от хоста

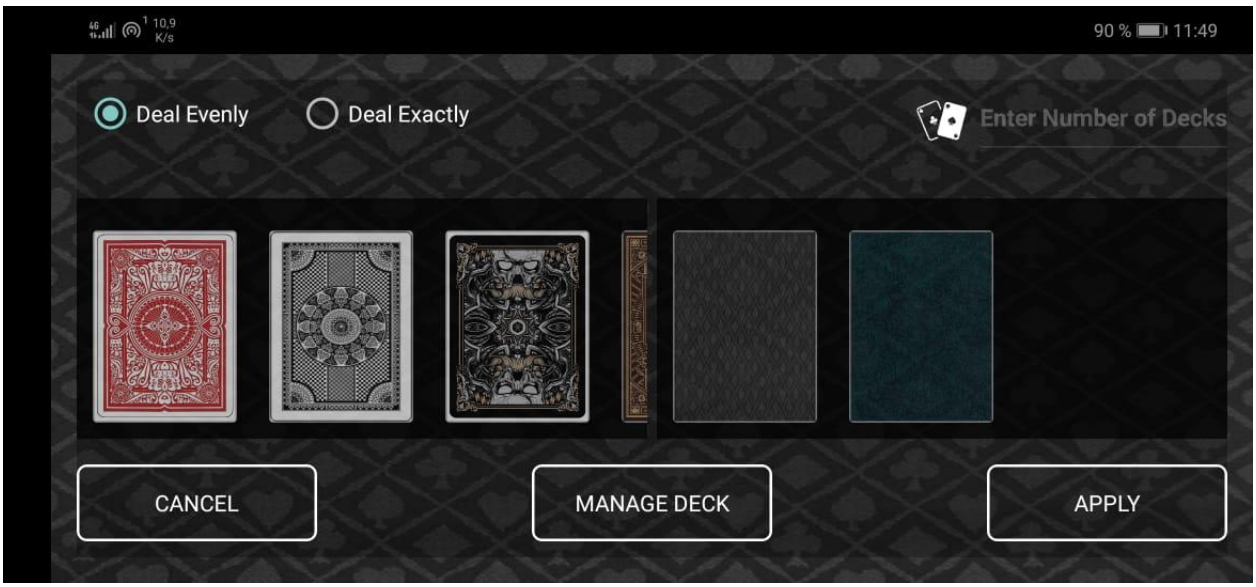

Рисунок 8 – Настройка игры

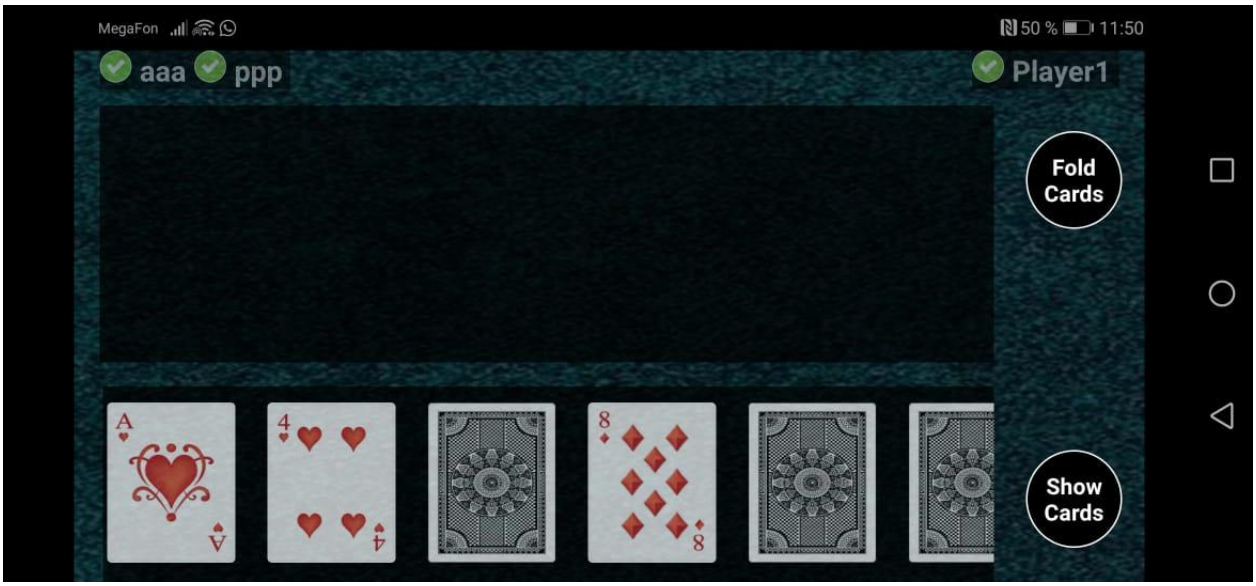

Рисунок 9 – Игра от клиента

Были проанализированы существующие аналоги и методы разработки, а также выбрана среда разработки. Для реализации поставленной задачи отлично подошла разработка с помощью AndroidStudio и языка программирования JavaScript. Такой выбор заметно упростил разработку проекта, так как в интернете имеется достаточное кол-во документации по разработке андроид приложений и JavaScript. Во время создания игры был полученный ценный опыт работы с этим средством разработки. В итоге была разработана и протестирована игра по схеме «Клиент – Сервер», что позволяет пользователям взаимодействовать в игре с разных Андроид телефонов в локальной или интернет сети. Такая возможность очень важна в наше время, потому что позволяет играть с другом или знакомым, не выходя из дома.

Библиографический список

1. Е.В.Непомнящих Проектирование пошаговых онлайн игр// Язык и социальная динамика. 2010. № 10-2. С. 33-35.Порядин А.Е., Сидоркина
2. Клюкин В.Э., Плотников В.Ю.Игровая программа-решатель маршрутных задач методом интеллектуального эвристического поиска типа клиент/сервер на языке swi-prolog http // Прикладная математика и фундаментальная информатика. 2014. № 1. С. 227-229.
3. Перлман С., Ван Дер Лан Р. Система для потоковой передачи баз данных, обслуживающих приложения реального времени, посредством потоковой передачи интерактивного видео// Инновационное образование. 2014. № 1 (8). С. 5-9.
4. Пушкин И.А. Разработка отказоустойчивой распределенной многопользовательской игры // Процессы управления и устойчивость. 2017. Т. 4. № 1. С. 631-635.
5. Claypool M. Network characteristics for server selection in online games// Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2009. № 4. С. 199-211.
6. Webb S. D., Soh S. Adaptive client to mirrored-server assignment for massively multiplayer online games//Proceedings of the 1st workshop on Network and system support for games. ACM, 2002. С. 53-57