Pengcheng Zhao

Mobile Phone Programming-
Multi-Player Mobile Phone Game
Based on Bluetooth

Bachelor’s Thesis
Information Technology

May 2010
DISCRIPTION

Date of the bachelor’s thesis
10 May 2010

Author
Pengcheng Zhao

Degree programme and option
Information Technology

Name of the bachelor’s thesis
Mobile Phone Programming-Multi-Player Mobile Phone Game Based on Bluetooth

Abstract

Mobile games are becoming more and more popular and the industry is increasing quickly. However, majority of these mobile games are single player games. In this thesis, I developed a multi-player game.

One of the key points in a multi-player game is the communication between mobile phones. There are many ways can achieve this, such as Infrared Ports, Wi-Fi, and Bluetooth technology. Of these three methods, Bluetooth is the most suitable one for my application.

In this thesis, I used Bluetooth API (JSR 82) to build a simple local wireless network to allow two mobile phones to exchange data. The game in this thesis is a rock-paper-scissors game which is introduced first. Next, the Bluetooth implementation is discussed.

Subject headings, (keywords)
Bluetooth, Java ME, Multi-Player Mobile Game, Bluetooth API, Nokia Remote Device Access

Pages
38 P.+App. 14

Language
English

URN

Remarks, notes on appendices

Tutor
Matti Koivisto

Employer of the bachelor’s thesis
Mikkeli University of Applied Sciences
ACKNOWLEDGEMENT

Thanks to my supervisor Matti Koivisto. He gave me a lot of wise advice about my thesis topic and writing. Without his constant supports, I could not finish this thesis in such a short period.

Also, I would like to give my thanks to all the people who helped me during I process my final thesis.

I would dedicate this thesis to my parents who gave everything to me.
### LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CDC</td>
<td>Connected Device Configuration</td>
</tr>
<tr>
<td>GAP</td>
<td>Generic Access Profile</td>
</tr>
<tr>
<td>CLDC</td>
<td>Connected Limited Device Configuration</td>
</tr>
<tr>
<td>GOEP</td>
<td>Generic Object Exchange Profile</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>HCI</td>
<td>Host Controller Interface</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared Radiation</td>
</tr>
<tr>
<td>IrDA</td>
<td>Infrared Data Association</td>
</tr>
<tr>
<td>ISM</td>
<td>Industrial, Scientific and Medical</td>
</tr>
<tr>
<td>Java ME</td>
<td>Java Micro Edition</td>
</tr>
<tr>
<td>JSR</td>
<td>Java Specification Requests</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>LANs</td>
<td>Local Access Networks</td>
</tr>
<tr>
<td>LMP</td>
<td>Link Manager Protocol</td>
</tr>
<tr>
<td>L2CAP</td>
<td>Logical Link Control and Adaptation Protocol</td>
</tr>
<tr>
<td>MIDP</td>
<td>Mobile Information Device Profile</td>
</tr>
<tr>
<td>OBEX</td>
<td>Object Exchange</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>RDA</td>
<td>Remote Device Access</td>
</tr>
<tr>
<td>RFCOMM</td>
<td>Radio Frequency Communication</td>
</tr>
<tr>
<td>SDAP</td>
<td>Service Discovery Application Profile</td>
</tr>
<tr>
<td>SPP</td>
<td>Serial Port Profile</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>2D</td>
<td>2 Dimensions</td>
</tr>
<tr>
<td>2G</td>
<td>Second Generation Wireless Telephone Technology</td>
</tr>
<tr>
<td>3G</td>
<td>Third Generation Wireless Telephone Technology</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 2.1: Image of two human bodies in mid-infrared ("thermal") light......................3
Figure 2.2: Infrared port in Nokia N 95........................................................................4
Figure 2.3: Wi-Fi logo..................................................................................................5
Figure 2.4: Bluetooth logo..........................................................................................5
Figure 2.5: Bluetooth architecture.............................................................................6
Figure 3.1: The Java platform.....................................................................................8
Figure 4.1: Start screen.................................................................................................10
Figure 4.2: Confirm screen...........................................................................................11
Figure 4.3: Result screen.............................................................................................12
Figure 4.4: The states and transitions for a MIDlet....................................................13
Figure 5.1: Package structure......................................................................................18
Figure 5.2: CLDC+MIDP+Bluetooth architecture diagram......................................18
Figure 6.1: “Host” or “Client” selection screen..........................................................26
Figure 6.2: Host's welcome screen.............................................................................27
Figure 6.3: Client's welcome screen...........................................................................27
Figure 6.4: Host chose “Scissors”................................................................................28
Figure 6.5: Client chose “Scissors”.............................................................................28
Figure 6.6: Surround Bluetooth devices list...............................................................28
Figure 6.7: The waiting screen and the alert message when client is sending data......29
Figure 6.8: Result on host's screen............................................................................29
Figure 6.9: Client's screen after send..........................................................................30
Figure 6.10: Nokia application installer......................................................................31
Figure 6.11: Selection screen......................................................................................32
Figure 6.12: Welcome screen.......................................................................................32
Figure 6.13: Device list on client screen.....................................................................33
Figure 6.14: Result on host screen.............................................................................33
# TABLE OF CONTENTS

1 INTRODUCTION ........................................................................................................ 1

2 COMMUNICATION BETWEEN MOBILE PHONES ........................................... 3
   2.1 Overview ............................................................................................................. 3
   2.2 Infrared Connectivity ...................................................................................... 3
       2.2.1 Infrared Radiation ................................................................................. 3
       2.2.2 Infrared Communication ................................................................. 4
   2.3 Wi-Fi Technology ......................................................................................... 5
   2.4 Bluetooth Technology ................................................................................. 5
       2.4.1 Introduction .......................................................................................... 5
       2.4.2 Bluetooth Architecture ................................................................. 6

3 PROGRAMMING ENVIRONMENT ........................................................................ 8
   3.1 Java Micro Edition ........................................................................................... 8
   3.2 Java Platform Micro Edition Software Development Kit 3.0 for Windows ....... 8
   3.3 NetBeans IDE 6.8 .......................................................................................... 9

4 IMPLEMENTING A SINGLE-PLAYER GAME ................................................ 10
   4.1 My Rock-paper-scissors Game ...................................................................... 10
   4.2 MIDlet ............................................................................................................. 12
   4.3 UI API ............................................................................................................ 13
       4.3.1 CommandListener ................................................................................ 13
       4.3.2 UI API Class Used in This Game ....................................................... 14
   4.4 Class GameCanvas ....................................................................................... 14
   4.5 Thread ............................................................................................................ 14
   4.6 Runnable ......................................................................................................... 15
   4.7 Graphics .......................................................................................................... 15
   4.8 Random ........................................................................................................... 16

5 BLUETOOTH PROGRAMMING WITH JAVA ME ....................................... 17
   5.1 Java APIs for Bluetooth Wireless Technology (JSR 82) ......................... 17
       5.1.1 Overview ............................................................................................... 17
       5.1.2 Architecture .......................................................................................... 17
   5.2 Discovery ........................................................................................................ 19
       5.2.1 Class DiscoveryAgent .......................................................................... 19
1 INTRODUCTION

Mobile games are the games played on the mobile devices such as PDAs, mobile phones or hand-held computers. They have a huge market. The first game that was pre-installed into a mobile phone was Snake in selected Nokia models in 1997. Nowadays, almost every mobile device carries several mobile games. Like the computer games industry, the mobile games industry has become more and more profitable. The total global revenues from the mobile games were estimated at 2,600 million USD in 2005. [1]

Java is the most common platform to develop mobile games. However, some mobile manufacturers, such as Nokia, Motorola, Sony, have developed their own platforms only for their products.

There are many types of mobile games, such as action, adventure, role-play, strategy, sports, puzzle, and so on. They can be played not only on a single player mode but also on a multi-player mode.

One of the key questions in multi-player games is communication between players (mobile phones), which can be achieved, for example, by the following methods:

- Infrared
- GPRS
- Wi-Fi
- Bluetooth

In my study, I have chosen Bluetooth technology as the communication method.

The aim of the study is to get familiar with Bluetooth technology on mobile phones and utilize it to implement a multi-player game. I will develop a multi-player rock-paper-scissors game.

The structure of the study is as follow:
In Chapter 2, I will introduce several methods which can be used in the communication between mobile phones. The emphasis will lie on Bluetooth technology.

In Chapter 3, I will briefly discuss the Java Micro Edition which I will use to develop my application.

In Chapter 4, I will present the rock-paper-scissors game part of my application.

Chapter 5 will illustrate the Bluetooth API and communication programming.

In Chapter 6, I will introduce the Rock-paper-scissors game application with Bluetooth implemented and the structure of it.

In Chapter 7, I will make a conclusion about this thesis. This chapter will also discuss the problems and difficulties I faced with during my study. Then I will discuss what kind of further work can be done in the future.
2 COMMUNICATION BETWEEN MOBILE PHONES

2.1 Overview

In this thesis, communication between mobile phones, means exchanging data between mobile phones in multi-player games. There are several wireless methods we commonly use—communication network, Infrared, Wi-Fi, and Bluetooth.

Communication networks include 2G and 3G services nowadays. They are always used in online games which allow a huge number of players play together at the same time. This thesis just concentrates on local multi-player games which only connect two mobile phones. Therefore, in this chapter, I will mainly introduce the other three communication methods mentioned above.

2.2 Infrared Connectivity

2.2.1 Infrared Radiation

Infrared radiation (IR) is a kind of electromagnetic radiation with a wavelength between red end of visible light and microwaves. We almost cannot detect any infrared portion of the electromagnetic spectrum from outer space. Figure 2.1 shows the infrared radiation from two human bodies. [2]

![Image of two human bodies in mid-infrared ("thermal") light](image)

**Figure 2.1:** Image of two human bodies in mid-infrared ("thermal") light [20]

The infrared part of the electromagnetic spectrum covers the range from roughly 300GHz (1mm) to 400 THz (750nm). It can be divided into three parts [3]:

- Far-infrared, from 300GHz (1mm) to 30 THz (10 μm). The lower part of this range may also be called microwaves.
• Mid-infrared, from 30 to 120 THz (10 to 2.5 μm). Hot objects can radiate strongly in this range.
• Near-infrared, from 120 to 400 THz (2,500 to 750nm). Physical processes that are relevant for this range are similar to those of visible light.

2.2.2 Infrared Communication

In infrared communication, infrared band is used as the transmission medium. The infrared devices employed in mobile phones are mostly for short-range wireless communication. The red rectangle in Figure 2.2 shows Nokia N 95’s infrared port. There are two domain standards for short-range infrared communication. One is IrDA standard and the other is IEEE 802.11. The later one is designed for wireless LANs and supports both radio and optical communication. [4] IrDA is a non-profit international organization which locates on California. It produces interoperable, low cost infrared data transmission standards. Many mobile phones made by Japanese manufacturers support IrDA standard. The developer can be registered as a member on IrDA official website and order the specifications or buy publications about IrDA standards. [21]

Figure 2.2: Infrared port in Nokia N 95 [5]

Compared with other communication methods, one of the disadvantages is that the infrared port should be exactly directed at the other port. Another drawback is its bad performance on anti noise and interferer. Light sources around the infrared devices, such as sunlight, incandescent lighting and fluorescent lighting, can interfere with the infrared communication. [22] Furthermore, infrared radiation cannot penetrate solid objects.

Although some flaws exist, many types of mobile phones are implemented with infrared ports for simple design, low cost and good data transmission rate.
2.3 Wi-Fi Technology

Wi-Fi is a brand registered by Wi-Fi Alliance. Figure 2.3 below is their logo. Wi-Fi uses radio frequency as the transmission medium. It is based on IEEE 802.11 standards. [23] Wi-Fi now includes four generations which are referred as 802.11a, 802.11b, 802.11g, and 802.11n. [24]

Compared with Bluetooth, Wi-Fi has a higher data rate which could reach theoretically 240Mbps for 802.11n. Also, it can achieve 100 meters. As a consequence, Wi-Fi consumes more power than other protocols. [25]

![Wi-Fi logo](image)

**Figure 2.3:** Wi-Fi logo [6]

Wi-Fi is widely used in computers wireless networks. Now more and more mobile phones are Wi-Fi certified. Users can create a peer-to-peer network without an access point. However, if there are more than three users, it is better to use an access point to create a Wi-Fi network. [26]

2.4 Bluetooth Technology

Bluetooth is the common way to connect multiple mobile phones together in a multiplayer game. In my game application, I will use Bluetooth as well.

![Bluetooth logo](image)

**Figure 2.4:** Bluetooth logo [7]

2.4.1 Introduction

Bluetooth wireless technology is designed for the replacement of cable connection
between digital devices. Theoretically, Bluetooth can cover minimum 10 meters or 100 meters which depend on the different device classes. 10 meters range Bluetooth devices are more familiar on mobile phones. The Bluetooth logo shown in Figure 2.4 can be found in the devices with Bluetooth. [8]

The key features of Bluetooth are low cost and low power consumption. But its data transmission rate is low. [25] Compared with the relatively high power consumed Wi-Fi, Bluetooth is more suitable for communication between mobile phones, because mobile phones have limited power supply, and it does not require high data rate.

2.4.2 Bluetooth Architecture

The following Figure 2.5 shows the basic layers and application framework of Bluetooth system.

![Bluetooth Architecture Diagram](image)

**Figure 2.5: Bluetooth architecture [29]**

The responsibilities of the layers in this stack are as follows [9] [28]:

- The radio layer is the physical wireless connection. Bluetooth communicates in the unlicensed ISM band. To avoid interference with other devices, frequency hopping is applied. It hops from one channel to another, up to 1600 times a second.
• The baseband layer is responsible for controlling and sending data packets over the radio link.

• The LMP is responsible for controlling and negotiating Bluetooth connection between devices.

• Between LMP and L2CAP, there is the Host Controller Interface (HCI), which is not shown in the figure above. It is the dividing line between software and hardware. The L2CAP and layers above it are currently implemented in software, and the LMP and lower layers are in hardware. The HCI provides a command interface to baseband and link manager.

• L2CAP stands for logical link control and adaptation protocol. It supports higher level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information.
3 PROGRAMMING ENVIRONMENT

3.1 Java Micro Edition

In my application, I used Java as the programming language. To develop Java application for mobile phones, developer should use Java ME. Mobile phones are mobile devices with limited hardware resources, which means they cannot run complicated applications as personal computers can. Java ME is the Java version which is designed for those devices with limited resources.

Figure 3.1 below represents an overview of the components of Java ME technology and how it relates to the other Java Technologies. [12]

![Diagram of Java ME technology](image)

**Figure 3.1:** The Java platform [12]

CDC and CLDC in the figure above are the two basic configurations. CDC is short for Connected Device Configuration and CLDC stands for Connected Limited Device Configuration. In this thesis, the Bluetooth API is for CLDC, which will be introduced in Chapter 5.

3.2 Java Platform Micro Edition Software Development Kit 3.0 for Windows

Java ME Platform SDK is a developing tool for mobile applications. By implementing
it, the developers can simulate and debug their applications on computers. The former edition is known as Java Wireless Toolkit 2.5.2_01 for CLDC. SDK 3.0 includes all the functions of this former edition. It integrates CLDC, CDC and Blu-ray Disc Java (BD-J) technology into one SDK. [13] Unfortunately, in this development kit 3.0, I did not figure out how to simulate applications with Bluetooth function. As a result, I just used this kit to test the game part of my application. For the part of Bluetooth, I used the Nokia Remote Device Access which I will discuss later.

3.3 NetBeans IDE 6.8

I used NetBeans IDE 6.8 as the development environment in my application. NetBeans IDE is a modular, standards-based integrated development environment (IDE), written in the Java programming language. The NetBeans project consists of a full-featured open source IDE written in the Java programming language, and a rich client application platform, which can be used as a generic framework to build any kind of application.

4 IMPLEMENTING A SINGLE-PLAYER GAME

In this chapter, I introduce a single player version of the rock-paper-scissors game. In this version there is no Bluetooth communication with other players but the user is playing against the mobile phone which randomly selects one of the possible alternatives (rock-paper-scissors). I first introduce the user interface of the game and then go to the implementation details. The multi-player version of the game will be explained later in Chapter 6.

4.1 My Rock-paper-scissors Game

First, there will be the start screen, which is shown in Figure 4.1. Press left key to choose “Rock”. Press down to choose “paper” and right key to choose “scissors”.

Figure 4.1: Start screen
After the user chooses one, the confirm screen will be displayed. Figure 4.2 is the confirm screen in the simulation.

**Figure 4.2:** Confirm screen

Press Menu->OK to confirm your choice. In other types of mobile phones, the “OK” command may be in the middle bottom of the screen. For example, as in Figure 4.2 above, the user choose the “scissors”. Then the result will be displayed on the screen as in Figure 4.3.
When the user presses the “OK” button, the computer will select one randomly. In this case, the computer chose “scissors” as well. It is a draw game. If the user wins, a text “You win!” will be displayed and otherwise “You lose!” will appear.

Now, the user can press Menu->Restart to start a new game. The start screen will come up. On the other mobile phones, the “Restart” button may appear on the right bottom of the screen.

Next, I will introduce all the classes and methods included in my game application.

4.2 MIDlet

Java Micro Edition is used for programming Java applications operated on the mobile phones. Every Java application must include a main class. MIDlet in Java ME is like
the main class in standard Java.

There are at least three methods in a MIDlet. They are `startApp()`, `pauseApp()`, `destroyApp()`. Figure 4.4 shows the states and transitions between them.

![Figure 4.4: The states and transitions for a MIDlet][10]

### 4.3 UI API

Using UI API, the developer can implement user interfaces in the application for MIDP. To use this API, the developer should import `javax.microedition.lcdui` first. There are four interfaces in this API, `Choice`, `CommandListener`, `ItemCommandListener`, and `ItemStateListener`. In this application, only `CommandListener` is implemented. [11]

#### 4.3.1 CommandListener

It is an interface which includes a method called as void `commandAction(Command command, Displayable displayable)`. When the mobile phone user presses a key on displayable d, the program will implement this method.
4.3.2 UI API Class Used in This Game

Canvas: The Canvas class is a base class for writing applications that need to handle low-level events and to issue graphics calls for drawing to the display.

Command: The Command class is a construct that encapsulates the semantic information of an action.

Display: Display represents the manager of the display and input devices of the system.

Displayable: An object that has the capability of being placed on the display.

Form: A form is a screen that contains an arbitrary mixture of items: images, read-only text fields, editable text fields, editable date fields, gauges, choice groups, and custom items.

Graphics: Provides simple 2D geometric rendering capability.


4.4 Class GameCanvas

Class GameCanvas provides the fundamental methods for creating a game interface. The methods in this class allow the developer to draw different kinds of graphics which can be displayed on the mobile phone screen. When GameCanvas is called, it will generate a dedicated buffer. The developer can modify this buffer by calling to the Graphics object(s). [30]

4.5 Thread

A thread is a thread of execution in a program. In Java programming, JVM allows multiple threads executed at the same time.
The following code would then create a thread and start it running:

```java
Thread t = new Thread();
t.start();
```

`sleeplong millis)`

By calling this method, it can cause the current thread to sleep certain milliseconds. The parameter `millis` tells the length of time you want the thread to sleep in milliseconds. [31]

**4.6 Runnable**

If the developer wants to execute the class by a thread, this interface should be implemented. There is only one method `run()` in this interface. When starting a new thread, the method `run()` will be automatically called. [32]

**4.7 Graphics**

This class Provides simple 2D geometric rendering capability. There are varied kinds of methods in this class. In my application, I only used three methods.

**FillRect**

It should be called in this formation: `fillRect(int x, int y, int width, int height)`. This method can fill the specified rectangle with the current color. X and Y indicate the location where the rectangle will be drawn. Width and height state the size of the rectangle. [33]

**SetColor(int RGB)**

This method sets the current color to the specified RGB values. If no other `setColor` method is called next, all subsequent rendering operations will use this current color.
The parameter *int RGB* in my application is in the form of hex. A hex number should start with “0x”. “0x0000ff” is the color of blue. “0x00ff00” represents red and “0xffff00” is yellow. [33]

**DrawString**

The formation of this method is `drawString(String str, int x, int y, int anchor)`. “str” is the string you want to draw. X and y locate the line where the string is drawn. Anchor tells how much blank should be left before the string. [33]

**4.8 Random**

Class `Random()` will generate a random number. In my application, I used `nextInt()` to return a randomly generated integer. Then I divided it by three and got the remainder of it. If the remainder is “0”, it means computer select “Rock”. “1” represents “Paper” and “2” is “Scissors”. [33]
5 BLUETOOTH PROGRAMMING WITH JAVA ME

5.1 Java APIs for Bluetooth Wireless Technology (JSR 82)

5.1.1 Overview

Java APIs for Bluetooth Wireless Technology (JSR 82) is designed for Connected Limited Devices Configuration (CLDC). In this API, it does not cover all the areas related to Bluetooth. It provides APIs for the following areas [16] :

1. Data transmissions
2. Four protocols (L2CAP, RFCOMM, SDP, Object Exchange protocol)
3. Four profiles (GAP, SDAP, SPP, GOEP)

By using this API, the developer can realize the following operations [16] :

1. Register services
2. Discover devices and services
3. Establish RFCOMM, L2CAP and OBEX connections
4. Conduct these activities in a secure fashion

To operate applications which include this API, the devices should meet the following minimum requirements [16] :

- At least 512K memory for Java 2 platform.
- The Bluetooth hardware
- Compatible with CLDC APIs or its superset APIs.

5.1.2 Architecture

There are two packages available in this API, the core Bluetooth API and OBEX API [17]. Both of them are optional packages, which indicates that the developer can import both packages or only one of them. The package structure is shown below in Figure 5.1.
Devices with MIDP+CLDC architecture are expected to be the most commonly used environment. Bluetooth API can work properly in this architecture. Figure 5.2 below shows the whole architecture.

![Figure 5.2: CLDC+MIDP+Bluetooth architecture diagram [16]](image)

According to the operations supported by this API, the functionality can be divided into three parts: discovery, communication, and device management. The next three sections will separately give details about how to archive these three functions. [16]
5.2 Discovery

Before two wireless devices start communication, they should know the information about the available devices around them, and what kind of services they can provide. In Bluetooth API, discovery is responsible for that.

5.2.1 Class DiscoveryAgent

Methods `startInquiry()` and `searchServices()` are included in this class. [34]

There are two parameters in `startInquiry()` method, `int accessCode` and `DiscoveryListener. AccessCode` indicates the inquiry type to be completed. Usually, we use General/Unlimited Inquiry Access Code (GIAC) as the inquiry type.

As for `searchServices()`, there are four parameters, `int[] attrSet`, `UUID`, `RemoteDevice` and `DiscoveryListener`. `UUID` tells the device to search the service with the specific `UUID`. After the device finds the certain service, it will retrieve the certain attributes according to the value of `int[] attrSet`. `RemoteDevice` indicates the device which will be searched for services. [34]

5.2.2 Interface DiscoveryListener

This interface contains four methods, `deviceDiscovered()`, `servicesDiscovered()`, `serviceSearchCompleted()`, `inquiryCompleted()`. They define what actions will be done when the device or service is discovered. Also, there are eight fields to show the states during the device inquiry and services search [35]:

`INQUIRY_COMPLETED, INQUIRY_ERROR, INQUIRY_TERMINATED, SERVICE_SEARCH_COMPLETED, SERVICE_SEARCH_DEVICE_NOT_REACHABLE, SERVICE_SERCH_ERROR, SERVICE_SEARCH_NO_RECORD, SERVICE_SEARCH_TERMINATED`.

Method `deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)` will be called when a device is found during an inquiry. When an inquiry is completed,
inquiryCompleted() will be called. Other two methods do the similar job during searching services.

5.2.3 Interface ServiceRecord

This interface records the characteristics of a Bluetooth service. In this thesis only getConnectionURL(int requiredSecurity, boolean mustBeMaster) method is called. Parameter requiredSecurity always is NOAUTHENTICATE_NOENCRYPT which indicates authentication and encryption is not needed on a connection to this service. If the value of mustBeMaster is true, this device should be the master in the connection. Otherwise, either master or client is OK. [36]

5.3 Device Management

5.3.1 Class LocalDevice

This class plays the role as the basic Bluetooth manager. It provides methods which can get access to and control the local Bluetooth device. In this thesis, only two basic methods are used.

- **getLocalDevice()**
  By calling this method, it will return the LocalDevice object.

- **setDiscoverable(int mode)**
  This method will set the Bluetooth device to a general or limited discoverable mode. To put the device into the general mode, the parameter mode should be DiscoveryAgent.GIAC. DiscoveryAgent.LIAC will make the device limited discoverable. The Bluetooth device can be only in limited mode for one minute. After the time is up, the device will be set to the previous mode. [37]
5.3.2 Class RemoteDevice

As the class LocalDevice, RemoteDevice represents a remote Bluetooth device. In my application, only one of the methods in this class is called, getFriendlyName(boolean alwaysAsk). If the parameter alwaysAsk is true, the device will contact the remote device for its name, otherwise, the name of the remote device will be returned without contacting it. [38]

5.4 Communication

The main purpose of getting local device and remote device is to establish a wireless connection between them, and to communicate with each other. This section will introduce how to establish a simple Bluetooth connection which allows the client to send data to the server.

One of the basic rules of Bluetooth communication is that the local device and remote device should use the same protocol. Bluetooth API supports the services with RFCOMN, L2CAP or OBEX protocols. RFCOMM is a kind of cable replacement protocol. This protocol is used in the application based on SPP. My application is based on SPP. So, I will emphasize the connection using SPP. [16]

5.4.1 Connection URL

The form of the URL is as follows [16] :

btspp://localhost:+UUID+server parameters

The number of parameters can range from zero to five. The parameters include name, master, encrypt, authorize and authenticate.

- Name should be in the form of “;name=”+text. Text can contain alpha, digit, “-” or “_”.
- Master should be in the form of “;master”+true/false.
- Encrypt should be in the form of “;encrypt=”+true/false.
• Authorize should be in the form of “;authorize=”+true/false.
• Authenticate should be in the form of “;authenticate=”+true/false.

5.4.2 Connection Establishment

The connection establishment requires co-operation between the server and the client. To establish a server connection, the following commands should be used [16] :

```java
StreamConnectionNotifier service =
(StreamConnectionNotifier)Connector.open("btspp://localhost:102030405060708090A1B1C1D1D1E100;name=SPPEx");
```

```java
StreamConnection con = (StreamConnection) service.acceptAndOpen();
```

The method `acceptAndOpen()` indicates that the server is ready to receive connection from a client. This method will block until a client connects to server.

After the server has established the connection, the server will wait for clients. The following commands can help establish a client connection [18] :

```java
url = service.getConnectionURL(
ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false);
conn = (StreamConnection) Connector.open(url);
```

5.4.3 Send Message

After the server and client establish the connection, they should open a data output and input stream before they can exchange messages. To send a message, the client can use `write()` method to send a number or a string. On the other hand, the server can call the method `read()` to receive the message from the client.

After the server and client have finished the communication, `close()` method in the `StreamConnection` can be called to close this connection.
5.4.4 Example Code

This example code is a part of the whole application. It only covers establishing connection. Processes of searching devices and services are not included in this example. The code in Appendices includes these processes.

First, here is the client code [16]:

class RFCOMMPrinterClient {
    private String serverConnectionString;
    RFCOMMPrinterClient(String server) {
        serverConnectionString = server;
    }
    public boolean printJob(String data) {
        OutputStream os = null;
        StreamConnection con = null;
        try {
            /*
            * Open the connection to the server
            */
            con = (StreamConnection) Connector.open(serverConnectionString);
            /*
            * Sends data to remote device
            */
            os = con.openOutputStream();
            os.write(data.getBytes());
            /*
            * Close all resources
            */
            os.close();
            con.close();
        } catch (IOException e2) {
            System.out.println("Failed to print data");
        }
    }
}
StreamConnectionNotifier service = null;
StreamConnection con = null;
InputStream is = null;
String serviceURL =
"btspp://localhost:102030405060708090A1B1C1D1E100;name=SPP Server1";
try {
    /*
    * Creates an SPP service record.
    */
    service = (StreamConnectionNotifier)
    Connector.open(serviceURL);
    /*
    * Add the service record to the SDDB and
    * accept a client connection.
    */
    con = (StreamConnection)service.acceptAndOpen();
    is = con.openInputStream();
    try {
        int ch;
        while ((ch = is.read()) != -1) {
            /* handle data received */
        }
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
} catch (IOException e) {
    System.out.println("IOException: " + e.getMessage());
    return false;
}
return true;
is.close();
/
* Close connection.
*/
con.close();
/
* Remove service record from the SDDB.
* Stop accepting connections.
*/
service.close();
} catch (IOException e) {
    System.out.println(e.getMessage());
}
6 IMPLEMENTING A BLUETOOTH BASED MULTI-PLAYER GAME

This chapter mainly discusses how I implemented Bluetooth communication into the rock-paper-scissors game which was mentioned in Chapter 4. First of all, I will introduce the simulation on Nokia Remote Device Access.

6.1 Simulation on Nokia Remote Device Access

6.1.1 Nokia Remote Device Access (RDA)

It is a service provided by Nokia. The website of this service is http://www.forum.nokia.com/Technology_Topics/Application_Quality/Testing/Remote_Device_Access/Instructions_for_RDA_Usage.xhtml. It allows the developer to get remote access to Nokia mobile devices in the Internet. Before using it, one should log in as a Nokia forum member. It is free of charge to register an account on Nokia forum. There is a detailed introduction to teach you how to use it. To put it in a nutshell, just click the “Remote Device Access” button on the website and then choose the available device. [19]

6.1.2 Simulation

In this case, I chose Nokia N73 and 6700 Slide as the test devices. N 73 will be the client and 6700 will be as server. When you first open RockPaperScissorsBT application, a selection list will appear as is shown in Figure 6.1 below:

![Figure 6.1: “Host” or “Client” selection screen](image.png)
After the user presses “OK”, it will display the welcome page. Figure 6.2 and Figure 6.3 are the welcome screens of the host and client.

![Figure 6.2: Host's welcome screen](image)

![Figure 6.3: Client's welcome screen](image)

Now, you can follow the introduction to choose “Rock”, “Paper” or “Scissors” by pressing “Left”, “Down” or “Right” button. Then press “OK”. In different types of devices, the “OK” button will be in different places of the screen. In Figure 6.4 and Figure 6.5, the host and client both chose “Scissors”.

![Figure 6.4: Host's screen](image)

![Figure 6.5: Client's screen](image)
After you press the “OK” button, the host will establish a connection and the client will start Bluetooth device search. When client finishes device search, all detected Bluetooth devices will be listed on the screen as shown in Figure 6.6.
You should select the host's Bluetooth device name. After that, the communication will begin. The client will send its game selection to the host. When communicating with the host, an alert will come out. Press “Yes” to allow this connection. Figure 6.7 is the alert screen shot.

Figure 6.7: The waiting screen and the alert message when client is sending data

As soon as the communication is finished, the result will be displayed on the host's screen. “Restart” button will appear on both the host's and client's screens. Press “Restart” will lead you to the welcome screen. Figure 6.8 is the host's screen. In this case, it is a draw game. Figure 6.9 is the client's screen after the communication.

Figure 6.8: Result on host's screen
6.2 The Structure of My Application

6.2.1 RockPaperScissorsBT MIDlet

One of the responsibilities of this MIDlet is displaying the “Host” and “Client” selection list. The application will create new RockPaperScissorsCanvas and enter it if CommandListener detect “OK” pressed. Or, if “Exit” is pressed, it will destroy the MIDlet to close this application.

6.2.2 Class RockPaperScissorsCanvas

The result of calling this class is that the welcome page will be drawn and displayed on screen by calling the Class Graphics. An integer variable m will store the player's selection. If the player chooses “Rock”, it will assign number 0 to int m. If “Paper” is chosen, number 1 will be assigned to m. If “Scissors” is selected, m will be 3. When “OK” is pressed, the program will enter the class Host or Client according to the choice of the user at “Host” and “Client” selection list.

6.2.3 Class Host

First of all, this class will help establish a Bluetooth server connection at the host side.
Then it will wait until the server receives a message from a client. An int variable message will record the value the device has received. When the communication is done, it will compare the player's selection with the received value. At last, this class will publish the game result.

6.2.4 Class Client

This class will be executed if the player chose “Client” at the beginning. It will first discover and list all the surrounding Bluetooth devices. When the inquiry procedure ends, the class will wait for the player's choice. Then it will try to get access to the service on the chosen device. After the connection is established properly, the client will send the player's selection int m to server.

6.3 Deploy Application on Mobile Phone

Mobile phones made by different manufacturers use different software to install Java applications. In this thesis, I just took Nokia mobile phones as an example. Firstly, Nokia PC suit should be installed on your computer. PC suit can be downloaded from Nokia products website. Then use the Nokia application installer which is included in PC suit to install the .jar file on Nokia mobile phones. The .jar file is generated in the dist file under the project file.

Figure 6.10 is the screen shot of Nokia PC suit application installer.

![Figure 6.10: Nokia application installer](image)
Figures from 6.11 to figure 6.14 are the screen shots of real mobile phones. The mobile phones in those pictures are Nokia 2700c (on the left) and Nokia N 73 (on the right).

Figure 6.11: Selection screen

Figure 6.12: Welcome screen
Figure 6.13: Device list on client screen

Figure 6.14: Result on host
7 CONCLUSIONS

The aim of my study was to get familiar with the Bluetooth technology on mobile phones and utilize it to implement a multi-player game. Finally, I successfully developed a multi-player rock-paper-scissors game. To make a conclusion, I will discuss the challenges I met and possible future work.

7.1 Challenges and Solutions

The critical part of my thesis was to figure out how to build a Bluetooth connection and make it work. The Bluetooth API specification helped me a lot. It clearly explains how Bluetooth communication works step by step. Following its instructions and some example code, I implemented Bluetooth communication into my game application successfully.

Another question that occupied a lot of my work time is the code troubleshooting. The point is that only modify one place at a time. If you change too many places once, you may never know which one mainly causes the trouble. Always, the consequence was that you had to set the whole part to the origin and debug from the very beginning.

7.2 Possible Future Work

7.2.1 More Than Two Players

In my thesis, Bluetooth communication was implemented into a relatively simple rock-paper-scissors game. After being familiar with developing Bluetooth application, the developer can implement Bluetooth into more complicated games which can be played by two or more players. In that case, the developer may deal with how to create connections between each players and make them exchange data.

7.2.2 Real Time Communication During Game

Rock-paper-scissors is a turn-based game. The server and client communication can
fulfill the game request. If it comes to a real time game such as shooting games or action games, which are more attractive and require real time data exchange during game, the server and client communication model used in my thesis will be unsuitable. Other Bluetooth protocols may support that kind of communication and break the limitation.

In my point of view, Bluetooth is an excellent medium to be utilized to develop mobile games, which can build a local network to allow a group of players play together——just like what we had done with our computers.
References

[1] <URL http://www.3g.co.uk/PR/May2005/1459.htm>
Department of Electrical and Computer Engineering
Boston University, Boston, MA 02215 USA.
[16] JSR82-spec_1.1.1.pdf
[28] <URL http://www.bluetooth.com/English/Technology/Works/Pages/Core_System_Architecture.aspx#8>
[35] <URL
[36] <URL
[37] <URL
[38] <URL
Appendix 1

/*RockPaperScissorsMIDlet*/

import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;

public class RockPaperScissorsMIDlet extends MIDlet
implements CommandListener{

    private static RockPaperScissorsMIDlet RPS;
    private Form rForm;
    ChoiceGroup list;
    private RockPaperScissorsCanvas rCanvas;
    private Command rExit,rOK;

    public RockPaperScissorsMIDlet(){
        RPS=this;
    }

    public void startApp() {
        rForm=new Form("RockPaperScissors");
        String[] ListNames = { "Host", "Client" }; 
        list = new ChoiceGroup("Please select.", Choice.EXCLUSIVE, ListNames, null);
        rOK = new Command("OK", Command.OK, 0);
        rExit = new Command("Exit",Command.EXIT,0);
        rForm.addCommand(rOK);
        rForm.addCommand(rExit);
        rForm.append(list);
        rForm.setCommandListener(this);
        Display.getDisplay(this).setCurrent(rForm);
    }
}
public void pauseApp() {
}

public static void quit(){
    RPS.notifyDestroyed();
}

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable s) {
    if (c.getCommandType() == Command.EXIT) {
        notifyDestroyed();
    }
    if(c.getCommandType()==Command.OK){
        String name = list.getString(list.getSelectedIndex());
        rCanvas = new RockPaperScissorsCanvas(Display.getDisplay(this), name);
        Command OKCommand=new Command("OK",Command.OK,1);
        Command ExitCommand = new Command("Exit",Command.EXIT,1);
        Command RestartCommand=new Command("Restart",Command.STOP,1);
        rCanvas.addCommand(OKCommand);
        rCanvas.addCommand(ExitCommand);
        rCanvas.addCommand(RestartCommand);
        rCanvas.start();
    }
}

}
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class RockPaperScissorsCanvas extends GameCanvas implements Runnable, CommandListener {

    private volatile boolean rNothing;
    private int m = 10, state1 = 0;
    public static String name, me;
    private Display rDisplay;

    public RockPaperScissorsCanvas(Display Display, String name) {
        super(true);
        this.name = name;
        this.rDisplay = Display;
    }

    public void start() {
        setCommandListener(this);
        rDisplay.setCurrent(this);
        rNothing = true;
        Thread t = new Thread(this);
        t.start();
    }

    public void stop() {
        rNothing = false;
    }

    public void run() {
        Graphics g = getGraphics();
    }
}
while(rNothing==true){
  input();
  draw(g);
  try {
    Thread.sleep(20);
  } catch (InterruptedException ie) { stop(); }
}

private void input() {
  int keyStates = getKeyStates();
  if(state1==0){
    if ((keyStates & LEFT_PRESSED) != 0){
      m=0;
    }
    if ((keyStates & DOWN_PRESSED) != 0){
      m=1;
    }
    if ((keyStates & RIGHT_PRESSED) != 0){
      m=2;
    }
  }
}

private void draw(Graphics g){
  g.setColor(0xffff00);
  g.fillRect(0, 0, getWidth(), getHeight());

  String help0="It's your turn";
  String help1="Left-Rock ";
  String help2="Down-Paper";
  String help3="Right-Scissors";
  g.setColor(0x0000ff);
  g.drawString(help0, 0, 0, 0);
  g.drawString(help1, 0, 20, 0);
}
g.drawString(help2, 0, 40, 0);
g.drawString(help3, 0, 60, 0);
g.drawString("You're"+name,0,80,0);

switch(m){
    case 0:
        g.setColor(0xff0000);
        String Rock="You: Rock";
        g.drawString(Rock, 0, 100, 0);
        break;
    case 1:
        g.setColor(0xff0000);
        String Paper="You: Paper";
        g.drawString(Paper, 0, 100, 0);
        break;
    case 2:
        g.setColor(0xff0000);
        String Scissors="You: Scissors";
        g.drawString(Scissors, 0, 100, 0);
        break;
    case 10:
        g.setColor(0xff0000);
        String Please="Please Input:";
        g.drawString(Please, 0, 100, 0);
}

flushGraphics();
}

public void OK_Pressed(){
    if(name=="Host"){
        state1=1;
        new Host(rDisplay,m);
    }
}
if (name=="Client"){
    state1=1;
    new Client(rDisplay,m);
}
}

public void Restart(){
    m=10;
    state1=0;
}

public void commandAction(Command c, Displayable s) {
    if (c.getCommandType() == Command.EXIT) {
        RockPaperScissorsMIDlet quit();
    }
    if (c.getCommandType()==Command.OK){
        OK_Pressed();
    }
    if (c.getCommandType()==Command.STOP){
        Restart();
    }
}
Appendix 3

/*Host Class*/

import javax.bluetooth.*;
import java.io.DataInputStream;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;

public class Host implements Runnable, CommandListener{

private StreamConnectionNotifier SCN;
private Form list;
private Display display;
private RockPaperScissorsCanvas rCanvas;
private int m;
private Command Restart;
public int message;

public Host(Display display, int Y)
{
this.display = display;
m = Y;
list = new Form(""");
Restart = new Command("Restart", Command.OK, 0);
Thread t = new Thread(this);
t.start();
}

public void run() {
UUID uuid = new UUID(0x0009);
LocalDevice localDevice;
try {
localDevice = LocalDevice.getLocalDevice();
localDevice.setDiscoverable(DiscoveryAgent.GIAC);
SCN = (StreamConnectionNotifier) Connector.open("btspp://localhost:" +

}
StreamConnection conn = SCN.acceptAndOpen();
DataInputStream in = conn.openDataInputStream();
message=in.readInt();
SCN.close();
}
}catch (Exception e) {
}

switch(m){
    case 0:
        list.append("You: Rock\n");
        break;
    case 1:
        list.append("You: Paper\n");
        break;
    case 2:
        list.append("You: Scissors\n");
        break;
}

if(message==0){
    list.append("Opponent: Rock\n");
}
else if(message==1){
    list.append("Opponent: Paper\n");
}
else if(message==2){
    list.append("Opponent: Scissors\n");
}

String Lose="You lose!";
String Win="You win!";
String Draw="Draw Game.....";
switch(m-message){
    case 0:
list.append(Draw);
break;
case 1:
    list.append(Win);
    break;
case -1:
    list.append(Lose);
    break;
case 2:
    list.append(Lose);
    break;
case -2:
    list.append(Win);
    break;
}
list.addCommand(Restart);
list.setCommandListener(this);
display.setCurrent(list);

public void commandAction(Command c, Displayable s) {
    if (c.getCommandType() == Command.OK) {
        rCanvas = new RockPaperScissorsCanvas(display, "Host");
        Command OKCommand = new Command("OK", Command.OK, 1);
        Command ExitCommand = new Command("Exit", Command.EXIT, 1);
        Command RestartCommand = new Command("Restart", Command.STOP, 1);
        rCanvas.addCommand(OKCommand);
        rCanvas.addCommand(ExitCommand);
        rCanvas.addCommand(RestartCommand);
        rCanvas.start();
    }
}
Appendix 4

/*Client Class*/

import javax.bluetooth.*;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;
import java.io.DataOutputStream;
import java.io.IOException;
import java.util.Vector;
import javax.microedition.lcdui.*;
public class Client implements Runnable,DiscoveryListener,CommandListener{

private LocalDevice local_device;
private int smessage;
private UUID uuid;
private DiscoveryAgent DisAgent;
private Vector devices;
private ServiceRecord service;
private RemoteDevice otherDevice;
private boolean deviceChosen;
DataOutputStream output;
private String url;
StreamConnection SC;
private Form Form;
private List DList;
private Command rRestart,rOK;
private Display display;
private RockPaperScissorsCanvas rCanvas;

public Client(Display display,int message){
    this.smessage=message;
    this.display=display;
    Form=new Form("RockPaperScissors");
DList = new List("Devices discovered", List.EXCLUSIVE);
rRestart = new Command("Restart", Command.OK, 0);
rOK = new Command("OK", Command.OK, 0);
deviceChosen = false;
devices = new Vector();
Thread t = new Thread(this);
t.start();
}

public void run() {
    try {
        local_device = LocalDevice.getLocalDevice();
        DisAgent = local_device.getDiscoveryAgent();
        local_device.setDiscoverable(DiscoveryAgent.GIAC);
        DisAgent.startInquiry(DiscoveryAgent.GIAC, this);
        uuid = new UUID(0x0003);
        while (!deviceChosen) {
            DisAgent.searchServices(new int[] { 0x0100 }, new UUID[] { uuid },
                                   otherDevice, this);
        }
    } catch (BluetoothStateException e) {
        e.printStackTrace();
    } catch (IOException f) {
        f.printStackTrace();
    }
}

public void deviceDiscovered(RemoteDevice RD, DeviceClass DC) {
    devices.addElement(RD);
}

public void inquiryCompleted(int discType) {
    switch (discType) {
    case INQUIRY_COMPLETED:
        if (devices.size() > 0) {
            // Code here...
        }
    }
DList.deleteAll();
for (int i = 0; i < devices.size(); i++) {
    try {
        RemoteDevice rd = (RemoteDevice) devices.elementAt(i);
        DList.append(rd.getFriendlyName(false), null);
    } catch (IOException e) {
    }
}
DList.addCommand(rOK);
DList.setCommandListener(this);
display.setCurrent(DList);
break;

case INQUIRY_TERMINATED:
    break;

case INQUIRY_ERROR:
    break;
}
}

public void serviceSearchCompleted(int transID, int respCode) {
    switch (respCode) {
    case SERVICE_SEARCH_TERMINATED:
        break;
    case SERVICE_SEARCH_ERROR:
        break;
    case SERVICE_SEARCH_NO_RECORDS:
        break;
    case SERVICE_SEARCH_DEVICE_NOT_REACHABLE:
        break;
    case SERVICE_SEARCH_COMPLETED:
        try {
            url =
service.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false);

    SC = (StreamConnection) Connector.open(url);
    output = SC.openDataOutputStream();
    output.writeInt(smessage);
    output.flush();
    output.close();
    SC.close();
    Form.deleteAll();
    Form.append("Send");
    Form.addCommand(rRestart);
    Form.setCommandListener(this);
    display.setCurrent(Form);
} catch (Exception e) {
    }
    break;
}
}

public void servicesDiscovered(int arg0, ServiceRecord[] servRecord) {
    service = servRecord[0];
}

public void commandAction(Command c, Displayable s) {
    if (c == rOK) {
        otherDevice = (RemoteDevice) devices.elementAt(DList.getSelectedIndex());
        Form.deleteAll();
        Form.append("Waiting...");
        display.setCurrent(Form);
        deviceChosen = true;
    }
    if(c==rRestart){
        rCanvas = new RockPaperScissorsCanvas(display, "Client");
        Command OKCommand=new Command("OK",Command.OK,1);
        Command ExitCommand = new Command("Exit",Command.EXIT,1);
Command RestartCommand = new Command("Restart", Command.STOP, 1);
rCanvas.addCommand(OKCommand);
rCanvas.addCommand(ExitCommand);
rCanvas.addCommand(RestartCommand);
rCanvas.start();
}