Current Mobile Message Editing Analysis and Adopting Customizing Facility in SMS

Mohammad Shamsul Arefin[1], Rezaul Karim[2], Galib Bin Aziz[1], Sheikh Md. Zakaria[1]
[1]Computer Science and Engineering Department, Chittagong University of Engineering & Technology, Bangladesh
[2]School of Computer Science and Engineering, University of Information Technology and Science, Bangladesh
e-mail: sarefin@cuet.ac.bd, galib02@yahoo.com, smzakaria@yahoo.com and pinnsele_of_success@yahoo.com

Abstract: *Modern days of mobile communication acquainted us with a technology of Short Messaging Service or shortly SMS. It helps the users to interact with other mobile subscriber using short message. This built in technology is not so efficient and offers limited options. The research area in this regard is to make it more efficient and flexible. In case of sending SMS, we do not have the opportunity to customize our message using different font styles. In this paper, the customizing facility has been introduced in case of SMS. The objective is to provide the user a freedom to select different types of font style. This application is very simple to install in every Java supported mobile handsets. The application takes about 200-250 KB storage, which is small enough to store in any java enabled mobile or other hand held devices. It can be regarded that this project work will create a new dimension in SMS service.*

Key Words: SMS, J2ME, mobile communication

1. INTRODUCTION

The Short Messaging Service (SMS) allows a handled device to generate and receive short messages. It was introduced in Europe in 1991 and became an instant hit. The Global System for Mobile Communications (GSM)—formerly the European, Standard for digital wireless- has supported SMS from the beginning. In USA, SMS was available initially on digital wireless networks built by BellSouth Mobility, PrimeCo and Nexttel among others. Short Messaging Service is an industry standard packet based wireless service that enables the transmission of alphanumeric messages between mobile subscribers and systems such as electronic mail, paging and voice mail systems.

In case of sending SMS there is no opportunity to customize the message using different font styles. In this paper a technique has been proposed to adopt this technique in SMS. The objective is to provide the user a freedom to select new types of font style such as Monotype Corsiva or Times New Roman etc. This package is very simple to install in every Java supported mobile handsets. The application takes about 200-250 KB spaces, which is small enough to store in any java enabled mobile or other hand held devices.

2. BACKGROUNDS AND PRESENT STATE

In SMS up to now there has been small improvement. As it is a new technology it needs some time to be established. But due to its endless popularity researches are going on to make it more efficient flexible and more users friendly. New attachments that have been added in this system have given this technology a boost. Today's wireless mobile communications devices offers text messaging services that enable short textual messages to be sent to the devices from any device that access to the service. There is no doubt that this service has provided a lot of convenience to us. SMS type of messaging is capable of sending a maximum of 160 characters on the control channel of a cellular telephone network.

The implementation of J2ME [2, 3, 4] in various mobile and hand held devices has successfully brings manifold advantages and flexibility to the mobile users. Day by day thousands of applications are being invented and are successfully implemented in different mobile sets. Some example of these types of applications includes survival dictionary, Download assistant, mobile safe, mp3 encoder (used in Nokia 3G sets), 3D phonebook etc. The motive behind these types of implementations is to provide support of multimedia facilities in such small devices. The small devices naturally contains very small amount of memory and thus supports very small amount of Java options. Therefore these types of packages are available as very small types of jar and jad files. All java supported mobile sets contain the built in compiler to run the corresponding jar and jad files.

The multimedia message service (provided by different mobile users) is the initial step to provide multimedia services to the mobile subscribers. But in our country this service is quite expensive and so, not so much popular yet.

The use of J2ME has made different kinds of multimedia service packages to be custom built by any Java programmers. Sometimes programmers use some tricky methods to reduce the cost while sending a multimedia message to the other end. This type of work is recently done by a group of students of Bangladesh University of Engineering and

ISBN 984-300-002131-3

Technology, which perform the encoding of Bangla fonts and to be sent just like any other normal SMS. The package is currently available in Internet and is only available to the AKTEL users.
In this paper, the same strategy has been employed but propose more flexibility to the mobile users in message editing. As it is known that J2ME does not support any kind of font styles the strategy of encoding the image fonts has been adopted.

3. METHODOLOGY

Due to the limitations of font styles in J2me programming and CLDC configurations we will represent different fonts in discrete .jpeg picture format to the user display only. But it is inefficient to send the pictures directly as it will use the MMS technology that will cost more than SMS. Therefore we have assigned a unique value for each of these pictures and send those values as normal text. The receiver inbox will receive these values, evaluate them and places appropriate picture fonts corresponding to those values. Thus the receiver is totally unknown of the techniques applied and will only receive the modified fonts that the sender intended to send him.
The sender has to write a message in the editor available in his mobile set after installing the jar and jad file. Then he can choose the fonts supported in that application from the menu. Fig.1 shows basic construction of this application.

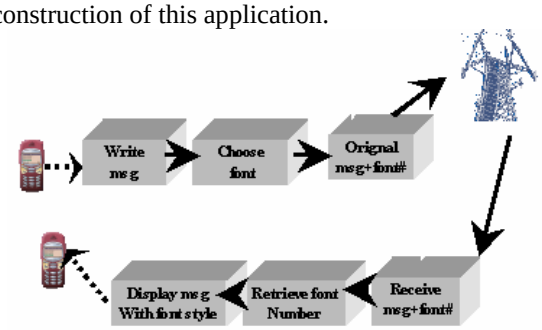

Fig. 1: Block Diagram of the Implementation

4. IMPLEMENTATION

4.1 Writing SMS

To write any message a form including a textfield has been declared. The Textfield will be used to write message in character and the form will be used to display it. For this *newTextField("","",159,TextField.ANY+TextField.INITIAL_CAPS_SENTENCE) used.* And other command like save command, back command also used in this form. Finally using *display.setCurrent(messageForm)* the has been displayed.

4.2 Getting message String

messageField.getString() has been used to get the string that has been written into the textfield. In order to display the written message in the form of changed font style the constructor of canvas class Preview has been called *new Preview(this, display, messaField.getString()).* Here the string of the message has been passed to the Preview class as a parameter. This message string will be used in the Preview class. Before sending this string to the Preview class an identifying character is enclosed at the end of the string length. It might be '*' or '#' or any character.

4.3 Sending Message String

First the string is taken into a variable say sms1 then it has been converted into the string object using *sendMessage.setPayloadText(sms1).* Finally the string message is sent using the connection *msgConnection.send(sendMessage).*

4.4 Displaying Message in different font styles

In the Preview class all the images of the font styles is loaded into an array. After the creation of the images these are atored in the slides[] array. Preview class gets the string parameter and copies it into another string. *st.getChars(0,st.length(),sms3,0).* Now it is easy for this class to break this string into string array. Now this array is divided and stored in an array. For each and every character a value is determined as a index for the array slides. It is done by using a simple equation *asc=ch-65* where 'ch' is the ascii value.

4.5 Receiving SMS

For receiving and displaying the short message accordingly i.e, with appropriate font styles selected by the sender we have constructed mainly three classes. The classes are named SMSreceive, receivedSMS and Prevr which perform the the smsport connection establishment, displaying module initializing and placing the Font pictures to their appropriate coordinates.

4.6 Message Port Connection

First we define a helper method that returns a MessageConnection. For creating a client Message connection we just call Connector.open(), passing a URL that specifies a valid WMA messaging protocol. In this particular application we have used the specific smsport which is retrieved by the getAppProperty() method. The MessageListener implements the Listener design pattern for receiving Message objects asynchronously; that is, without blocking while waiting for messages.

ISBN 984-300-002131-3

4.7 Initializing the display of the incoming message

The ReceivedSMS class is constructed to initialize the Prevr class. It is also used for passing the mainscreen parameter to the Prevr class to return to the Main Screen.

4.8 Displaying the message in the Canvas class

The operation of this class is almost the same like the Preview class defined for the sending preview purposes.

5. EEPERIMENTAL RESULT

5.1 Environmental setup

From the Toolkit first package of the software must be created. Which will produce the required .jar and .jad file of the package. This option will first compile the whole MIDlet, saves the project settings and then create the required application files.

5.2 Installation of the application

In the second step we need to install this newly created .jar and .jad file in the simulation tool kit. For this several steps are to be followed and when installation is finished applications settings is needed to be changed. Fig. 2, Fig. 3 and Fig. 4 depict the installation processes. Fig. 5 shows how to change the application settings not to interrupt the application at any condition.

Fig. 2: Install application

Fig. 3: Installing application (Cont.)

Fig. 4: Installing .jad file

Fig. 5: Changing Application settings

5.3 Launching the software:

After finishing the installation we can launch the software from the menu. Fig. 6 shows the launching process.

Fig. 6: Launching the software

5.4 Software interface:

In the main screen there are several options. From the create new options new message writing window comes out. Fig. 7 shows how to change the font styles or to save the message. Fig. 8 displays the canvas after changing the font styles.

Fig. 7: Write new message

Fig. 8: Message displayed in [sending mobile] Monotype font [sending mobile]

5.5 Recipient phone number entry:

To send new message from the menu send needs to be selected. A form will appear inquiring the receivers from number. Which is in this case +5550000. Fig. 9 shows the recipient phone number entry interface.

Fig. 9: Receivers phone no [sending mobile]

5.6 Receiving Procedure:

In the receiving end to receive this message the receiver needs to select the receive options. Fig. 10 depicts the receiving interface available in the receivers mobile. Fig. 11 shows the canvas with the corresponding font style sent by the sender.

ISBN 984-300-002131-3

Fig. 10: receive option selected

Fig. 11: Message received in changed [Receiving mobile] font style [Receiving mobile]

6. CONCLUSION

In this paper we proposed a new way to transmit SMS using different types of font style. For this elementary approach we have tried to keep things simple and small. Finally we have been succeeded to send SMS in two types of font style Monotype Corsiva and in Comic Sans. For these two types of font styles anyone can send SMS in the cost of the regular SMS. It is only a preliminary step in changing the font styles in mobile. Definitely following this path more works on this field can be done.

To provide large amount of font support we need to do some improvement here. We may need to draw them instead of using the picture. We will try to give a boost to install new type of font styles in its next version. We hope to give a new dimension in sending SMS by introducing animated SMS. Now we are providing limited font support. In our work we would try to include new properties that will support installing new type of font styles. We are also thinking about different blinking messages that are available now days as MMS/picture format. These types of formats can be easily converted to SMS format by adding some check bits in appropriate places, the strategy we have followed in this application. Furthermore, the user can modify the application to change the font sizes willingly. In summary, the package can be developed as any other text editing software available to use in our personal computers.

REFERENCES

[1] J. Keogh, *The complete Reference*, Tata McGraw-Hill, 2003

[2] K. Topley, *J2me in a Nutshell*, O'Reilly, March 2002

[3] M. Morrison, *Wireless Java with J2me*, Prentice Hall, 2003.

[4] V. Piroumian, *Wireless J2ME™ Platform Programming*, Mcgraw Hill, 2004.

[5] Sun Java, www.prenhall.com/deitel.com

[6] OsborneLimited, http://www.osborne.com

ISBN 984-300-002131-3