



PAN LOCALIZATION CAMBODIA OF IDRC

Khmer SMS TECHNICAL REPORT

Project Manager : Mr. CHEA, SOK HUOR

Lead Developer : Miss KHEM, SOCHENDA

Developer Team : Mr. NETH, SOVATHENA

Date : September 29, 2009

Table of Contents

I.	Introduction	1
II.	Literature Review	1
III.	Method.....	2
I.1.	Why j2me?.....	2
I.2.	Why microEWT?	2
I.3.	Khmer Script Analysis	2
I.3.i.	Types of Khmer Script.....	3
I.3.ii.	Rendering feature of Khmer Script.....	5
i.	Ligature.....	7
I.4.	Font Design.....	8
I.5.	Rendering Engine Development.....	9
I.5.i.	Creating display methods.....	9
I.5.ii.	Defining Khmer Character Cluster	14
I.5.iii.	Abbreviations	15
I.5.iv.	Algorithm approach.....	15
I.5.v.	Rendering Rule	16
I.5.vi.	Reordering	16
I.6.	Suggested Keypad Layout for Khmer Script	17
I.7.	Input Method using Two-Key press.....	19
I.8.	Interface Development	19
I.8.i.	LCDUI	21
I.8.ii.	microEWT	21
I.9.	Send and Receive text messages.....	22
I.9.i.	Using WMA 1.1.....	22
I.9.ii.	Using Push Registry	26
I.10.	SMS Storage.....	26
I.10.i.	Understanding Record Management System (RMS).....	26
I.10.ii.	Why RMS?	26
I.10.iii.	The use of RMS in KSMS.....	27
II.	Experiment Result	28
III.	Conclusion	29
	References.....	29

Table of Figures

Figure 1 Layers of Khmer script writing	2
Figure 2 Khmer Subscript rendering relative to its consonant	3
Figure 3 Khmer Subscript rendering relative to its consonant	4
Figure 4 Above vowels in layer 1 and 2.....	4
Figure 5 Left vowels in layer 3.....	4
Figure 6 Right vowels in layer 3	5
Figure 7 Right vowels in layer 3	5
Figure 8 Pre-base rendering.....	6
Figure 9 Post-base rendering	6
Figure 10 Above-base rendering	6
Figure 11 Below-base rendering	6
Figure 12 Two-Part base rendering	7
Figure 13 Ligature rendering.....	7
Figure 14 Subscript rendering	8
Figure 15 Khmer Bitmap Font	8
Figure 16 Khmer Bitmap Font and Special markers.....	9
Figure 17 Draw any bodies of Characters	9
Figure 18 Ligature composed of a consonant with vowel AA (ា).....	10
Figure 19 Ligature composed of a consonant with a part of vowel AU (ាវ).....	10
Figure 20 Ligature composed of ញ with vowel AA in case that ញ has a subscript.....	10
Figure 21 draw Ligature composed of Right subscript with vowel AA (ា)	11
Figure 22 draw Ligature composed of Right subscript with the right part of the vowel AU (ាវ).....	11
Figure 23 draw Left subscript, ceoung RO (រ).....	12
Figure 24 draw Right subscript	12
Figure 25 draw below subscript	12
Figure 26 draw subscript of NYO	13
Figure 27 draw Below vowels.....	13
Figure 28 draw Above vowels.....	13
Figure 29 draw Above Characters.....	14
Figure 30 How to draw Two-Part Vowel.....	14
Figure 31 generating reorder in KCC.....	16
Figure 32 generating reorder in KCC ្រ្រ្រ].....	17
Figure 33 Suggested Keypad Layout for Khmer Script.....	19
Figure 34 Structure of KSMS' interfaces.....	20
Figure 35 Main menu in KSMS	20
Figure 35 Alert and List using LCDU displayed in Nokia S40I.....	21
Figure 37 Generic Connection Framework (GCF).....	23
Figure 37 Components in WMA 1.1	23

I. Introduction

Number of mobile phones in Cambodia are increasing from day to day while there are only very few mobile phones (Khmer built-in) supported Khmer Script. Most of Cambodian users do not know English that means a constraint for them to use text message in English. Furthermore, if they want to compose Khmer SMS in none Khmer built-in mobile phone, they must use Latin script to write as Khmer sound phonetic. Yet it is another constraint if the users are unable to understand Latin script. Moreover, include not only students and those who can read and write Khmer correctly but there also exist many users who can only read and write Khmer very little. It is widely agreed that Khmer language can be promoted especially among young people. Since cellular phone has become popular, using SMS in Khmer script appears to be an effective way in promoting the language.

Now a day, the information technology is very up to date in the developed country. To enhance the capacity of information technology, the interaction from

Khmer SMS (KSMS) application development, a project of PAN Localization Cambodia (PLC), is aimed at providing Cambodian users an application to use SMS (Short Message Service) in Khmer Script. That means it is able to display Khmer Script correctly according to the feature of Khmer language. In addition, the application provides a friendly user interface which operates in most of cellular phones. The report will detail the techniques used during the research and development of KSMS and also present the result of the research.

II. Literature Review

There are some Khmer built-in cellular phones existing in Cambodian market. Recently, Nokia Company provides fifteen models of Khmer supported cellular phones [1] as in the table below.

Table 1 List of Khmer supported mobile phone from Nokia

N	Model
1	Nokia 7100 Supernova
2	Nokia 7070
3	Nokia 5130 Xpress Music
4	Nokia 5000
5	Nokia 2760
6	Nokia 2680 Slide
7	Nokia 2630
8	Nokia 2600 Classic
9	Nokia 1680 Classic
10	Nokia 1650
11	1661
12	2323
13	2330
14	5030 XpressRadio
15	2700 Classic

Though they are Khmer built-in cellular phone, they still have some mistakes to display Khmer script correctly, i.e. some Khmer words are displayed by overlapped characters; some characters are displayed in wrong position due to the complex characteristic of Khmer Script; some words cannot be typed at all.

Khmer Thmey (www.khmerthmey.com) has developed a Khmer SMS application named **Osen** [2]. This application cannot be used with most of cellular phones while it is specific used with only Nokia series 60. It provides two options for sending message as text or image. Its font is displayed similar to Limon font in Computer, i.e. there is no ligature occurs when a consonant meets a vowel like vowel AA (ា), vowel OO (ោ), or vowel AU (ៅ).

III. Method

To develop Khmer SMS application running on mobile devices, we use j2me (java 2 Micro Edition) and microEWT, which will be explain later, to enable Khmer script in mobile phone.

1.1. Why j2me?

J2ME (Java 2 Micro Edition), a Java platform for mobile devices and embedded systems, is the version of the Java language optimized for mobile applications [3]. It provides a robust and flexible environment for application running on mobile and other embedded devices [4]. Consequently, most major manufacturers offer devices with support of J2ME technology, i.e. presently most mobile devices in use around the world are almost J2ME supported. Hence, the choice of using j2me to develop mobile application has the opportunity to operate in most of mobile devices.

1.2. Why microEWT?

microEWT, an open source project under GNU General Public License (GPL), provides a graphical, event-driven interface library for j2me application [5]. In addition, it has an important feature which is the ability to render text with bitmap fonts. This leads our application to be capable of displaying Khmer font, which will be detailed later in font design, in mobile devices.

1.3. Khmer Script Analysis

Prior to the development of rendering engine, Khmer script needs to be analyzed. Khmer script is in type of complex script which has complex rendering position; it can be written into 5 layers as illustrated in the figure below.



Figure 1 Layers of Khmer script writing

I.3.i. Types of Khmer Script

Khmer script can be grouped into 6 main categories: Consonant, Subscript, Dependent Vowel, Consonant Shifter, Independent Script, and Various Sign.

a. Consonant

Consonant, the main character of every Khmer syllable, stands in layer 3 of Khmer writing structure. It is an independent character; however, it can be written with other types of Khmer script such as subscript and/or dependent vowel. Khmer script has 35 consonants which can be divided into three series. First, the consonants such as ង, ម, រ, យ, វ, ព្យ lack light voice whereas the second series such as ហ្ល, ស្រ, ប, អ lack heavy voice. The rest of these are considered into the third series. The first series including ប can be used with MUUSIKATOAN (◌[◌]); the second can be used with TRIISAP (◌[◌]); while the third cannot be used with MUUSIKATOAN or TRIISAP at all.

b. Subscript

Subscript is a character which is owned by a consonant. That means every consonant has its own subscript. Subscripts can be divided into three groups according to their rendering position compared to their main consonant as illustrated in figure 2. They are (1) Below subscript, (2) Left subscript, and (3) Right subscript. Left and right subscript stands in from layer 3 to layer 4 or till layer 5, while below subscript stands only in layer 4.

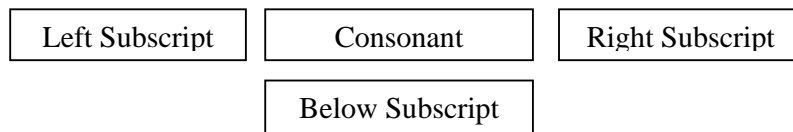


Figure 2 Khmer Subscript rendering relative to its consonant

c. Consonant Shifter

Consonant shifters are the two Khmer signs: (1) MUUSIKATOAN (◌[◌]) and (2) TRIISAP (◌[◌]). They are located in layer 2 over some of Khmer consonants. They are used to shift between *qakhoosaq* sound and *khoosaq* sound. *qakhoosaq* sounds –a while *khoosaq* sounds –o .

- 1) MUUSIKATOAN stands over consonant of the first series (ង, ម, រ, យ, វ, ព្យ) to shift to sound –a, voiceless. For instance, ង[◌], ម[◌], រ[◌], យ[◌], វ[◌], ព្យ[◌].
- 2) TRIISAP stands over consonant of the second series (ហ្ល, ស្រ, ប, and អ) to shift to sound –o, voiced. For instance, ហ្ល[◌], ស្រ[◌], ប[◌], អ[◌].

d. Dependent Vowel

Dependent vowels are Khmer characters which cannot stand alone. They must depend on a consonant. Due to their rendering position with main consonant or subscript, dependent vowels can be classified as five groups: (1) Below vowel, (2) Left vowel, (3) Right vowel, (4) Above vowel, and (5) Two-part vowel.

- i. **Below vowels** can stand only below main consonant or subscript. Hence, they can locate in layer 4 or layer 5. There are only 3 characters of this type: ្ក, ្ខ, ្គ. The figure below illustrates the rendering position of this type.

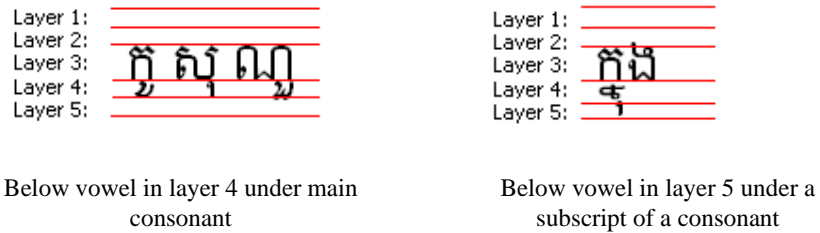


Figure 3 Khmer Subscript rendering relative to its consonant

- ii. **Above Vowels** can stand only above main consonant or consonant shifter. They locate in layer 1 only if there is a consonant shifter stays in layer 2 over the consonant in layer 3. In Khmer script, there are 5 above vowels such as ្ក, ្ខ, ្គ, ្ឃ, and ្ង.

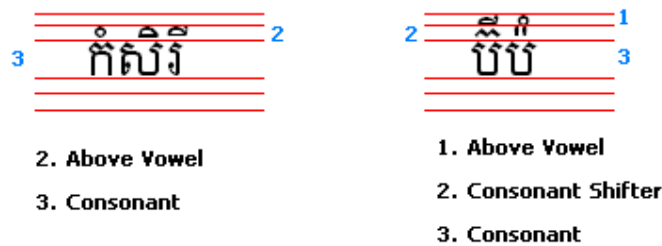


Figure 4 Above vowels in layer 1 and 2

- iii. **Left Vowels** can stand on the left side of main consonant or left subscript. Only 3 characters of Khmer script are left vowel such as ្ក, ្ខ, and ្គ.

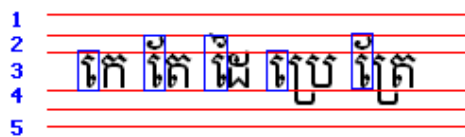


Figure 5 Left vowels in layer 3

- iv. **Right Vowels** stand on the right side of main consonant or right subscript. They locate in layer 3 with the consonant as well. They are ័, ័័, and ័័.

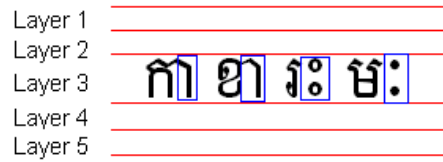


Figure 6 Right vowels in layer 3

- v. **Two-part Vowels** stand on both left and right/above side of main consonant. Generally the left part of them is vowel E (័). They are ័័, ័័័, ័័័, ័័័, ័័័, and ័័័. In that, vowel OO (័័) and AU (័័) can form a ligature, which will be explained later, whenever they meet a consonant.

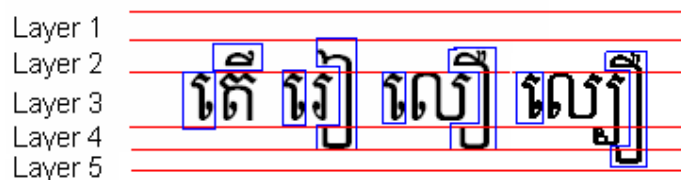


Figure 7 Right vowels in layer 3

e. Independent Script

Independent Scripts are referred to all Khmer characters that can stand alone without depending on other types of Khmer script. They are including Khmer Independent Vowels, Khmer digits, Khmer numeric symbols for divination lore, currency symbol, etc.

f. Various Sign

Various signs are Khmer dependent scripts such as punctuation and CEOUNG sign (័). They cannot be used with consonant shifter. Thus, they can stay in layer 2 or layer 4.

I.3.ii. Rendering feature of Khmer Script

As the types of Khmer Script mentioned above, Khmer Script rendering can be defined into six basic features: Pre-base, Below-base, Above-base, Post-base, Two-Part base and ligature.

a. Pre-base feature

Pre-base feature is a feature that a character needs to be placed before or on the left of another character. It is applied to Left vowels and Left subscripts. Consequently, reordering of characters' position is occurred.



Backing Store	Pre-base
	

Figure 8 Pre-base rendering

b. Post-base feature

Right Vowels and Right subscript have rendering position in Post-base feature, i.e. they locate on the right side of consonant. Furthermore, Right Vowel--Sign AA (o1)--will render to form ligature that will be explained below.



Backing Store	Post-base
	

Figure 9 Post-base rendering

c. Above-base feature

Above Vowels, Consonant Shifter and Various Signs (except ๑) have rendering position in Above-base feature.

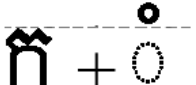

Backing Store	Above-base
	

Figure 10 Above-base rendering

d. Below-base feature

Below Vowels, Below Subscripts, and Various sign 17D2 (๑) have rendering position in Below-base feature.

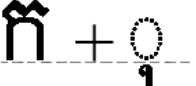

Backing Store	Below-base
	

Figure 11 Below-base rendering

e. Two-Part base feature

This feature can be found when a character has either Pre-base or Post-base feature. Two-Part vowel is applied with this feature. Moreover, two of Two-Part Vowel-- វិ and វី --also render to form ligature which will be explained soon.

Backing Store	Two-Part Base

Figure 12 Two-Part base rendering

i. Ligature

Khmer Unicode has some ligatures that are new shapes derived from combination of two characters. It means that two characters rendered by changing their shape to a new one. As general, Ligature of Khmer Unicode can be found only if consonant or right subscript meets left vowel ិ or two-part vowel: វិ, វី. This feature is illustrated as the figure below.

Backing Store	Ligature

Figure 13 Ligature rendering

Note: Since Khmer Unicode has no unique code for each subscript, U+17D2 combines with each Unicode of consonant are used to identify subscript, i.e. subscript can be seen when U+17D2 meets a consonant. For example, 17D2 + ក → ក្រ, 17D2 + ង → ង្រ.



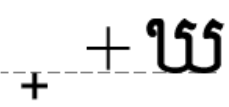
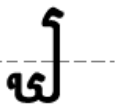
Backing store coeng u17D2 + u1780 	Below-base 
Backing store coeng u17D2 + u1783 	Post-base 

Figure 14 Subscript rendering

I.4. Font Design

A bitmap font is preferred to use in terms of Khmer font in KSMS. It contains many characters: all usable Latin characters, all Khmer characters, and additional shapes resulted from rendering, i.e. ligature. These characters are arranged orderly in one line as a horizontal image in PNG format. In that, we initially choose shapes of font named KhmerKep, one of Khmer Unicode fonts developed by PAN Localization Cambodia (PLC), with size of 16 pt to draw each character for the bitmap font. This is done because of two reasons:

- There are many screen sizes of mobile such as 128x160 px, 176x220 px, 240x320 px, 800x480 px, etc. However, the screen size 240x320 px is chosen since it is the popular one used by most mobiles development. Thus, the selected font size should be 16 pt which is the best for displaying, i.e. its size is not too big and not too small on the screen size.
- KhmerKep has small and thin body of shapes among all Unicode fonts belong to PLC. As a result, it is the best for displaying Khmer script on mobile screen.

As in figure 2, the Khmer font is showed in four lines in order to see all characters; however, the real font image is only a single line.

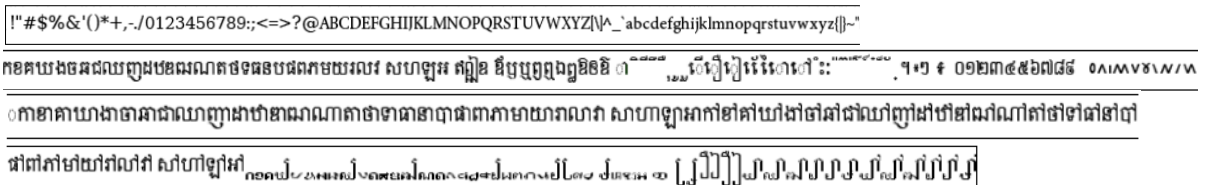


Figure 15 Khmer Bitmap Font

The image is transperance while using black color for each character which is arranged horizontally with special markers, in size of 1 pixel, on the above edge as illustrated in the figure below.

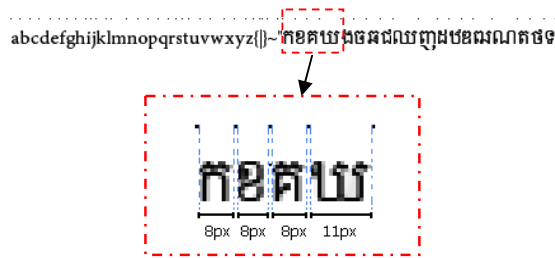


Figure 16 Khmer Bitmap Font and Special markers

The special markers are useful for the application to define each character’s position and its width in the whole image.

1.5. Rendering Engine Development

Since Khmer script has complex writing system, rendering needs to be done in order to show Khmer script on mobile’s screen. Rendering is the process of KSMS application works to display Khmer characters properly as their glyphs or ligatures in the correct positions base on the characteristic of Khmer language writing. It generates reordering or changing shapes to another one. Glyph is an element of writing, i.e. ក, ខ, ង, ៀ, ័, ុ, etc, while Ligature is a combination of two glyphs, i.e. ក + ង → កង.

The development of Rendering Engine involves four main tasks: (1) creating display methods, (2) defining KCC (Khmer Character Cluster), (3) rendering rule, and (4) generating reorder.

1.5.i. Creating display methods

As mentioned above, Khmer is a script with complex rendering. In order to display Khmer characters, we need to create methods which responsible to display each rendering feature. These methods are as the following:

- drawBoday(): It is used to draw whatever characters of Khmer script and Latin Script in case of the character independent or it stands alone while it cannot render with other. Dotted circle is used with dependent vowel to state that it is invalid combination. Figure 16 illustrates the display of this method.

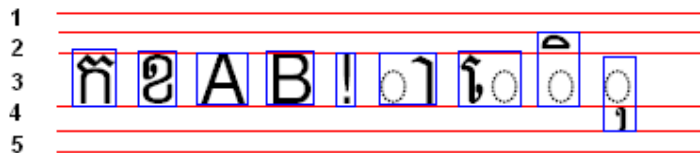


Figure 17 Draw any bodies of Characters

- drawLigatureAA(): It is used to draw ligature, figure 17, which is the combination between a consonant and the vowel AA (ង).

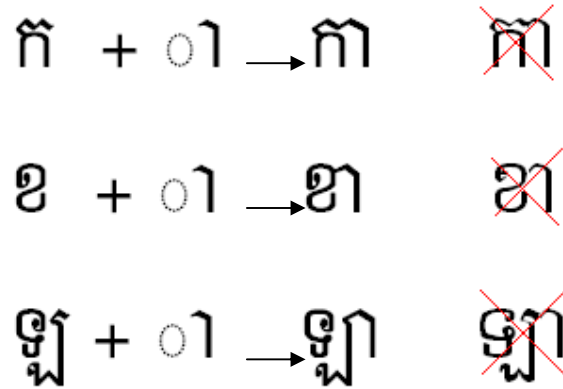


Figure 18 Ligature composed of a consonant with vowel AA (ា)

• drawLigatureAU(): It is used to draw ligature which is the combination between a consonant and the right part of vowel AU (ៅ). See the figure below.

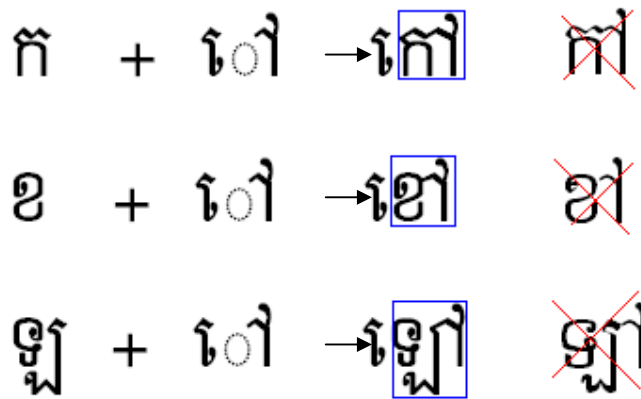


Figure 19 Ligature composed of a consonant with a part of vowel AU (ៅ)

• drawLigatureAAofNYO(): It is used to draw the ligature, the combination of consonant NYO (ញ) with vowel sign AA (ា), in case that consonant NYO (ញ) has a subscript stays under it.

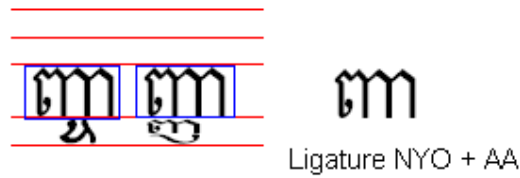


Figure 20 Ligature composed of ញ with vowel AA in case that ញ has a subscript

•drawLigatureAUofNYO: It is used to draw the ligature which is the combination between consonant NYO (ញ) and the right part of vowel sign AU (្រ្រ). However, It is used only when the consonant NYO (ញ) has a subscript stays under it.

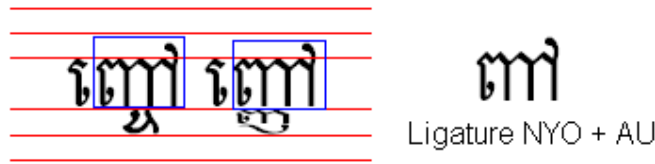


Figure 20: Ligature composed of ញ with vowel AU in case that ញ has a subscript

•drawRightSubscriptAA(): It is used to draw a ligature which is a combination between a right subscript and the vowel sign AA (្រ).

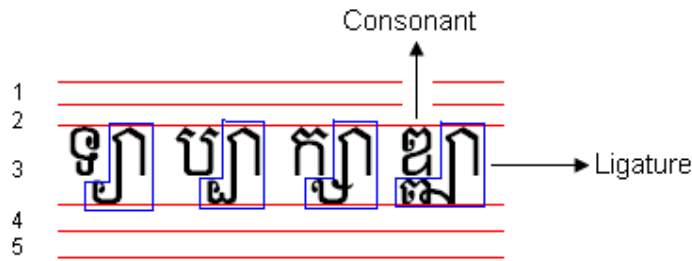


Figure 21 draw Ligature composed of Right subscript with vowel AA (្រ)

•drawRightSubscriptAU(): It is used to draw a ligature which is a combination between a right subscript and the right part of the vowel sign AU (្រ្រ).

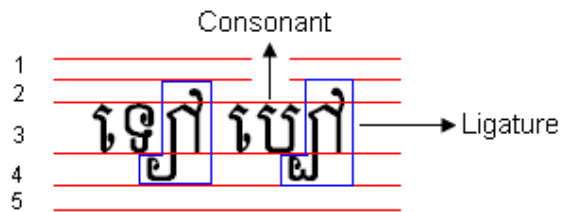


Figure 22 draw Ligature composed of Right subscript with the right part of the vowel AU (្រ្រ)

•drawLeftSubscript(): It is used to draw a subscript, ceung RO, which its rendering position is on the left of consonant. There are two possible cases occurred as illustrated in the figure below.

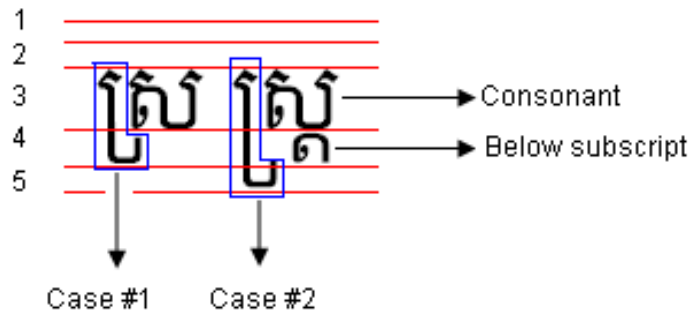


Figure 23 draw Left subscript, ceoung RO (ស)

Case #1: ceoung RO stays in layer 3 and layer 4 when there is no below subscript renders with.

Case #2: ceoung RO stays long from layer 3 till layer 5 because there is a below subscript renders with it.

• drawRightSubscript(): It is responsible to display Right subscript that render with main consonant.

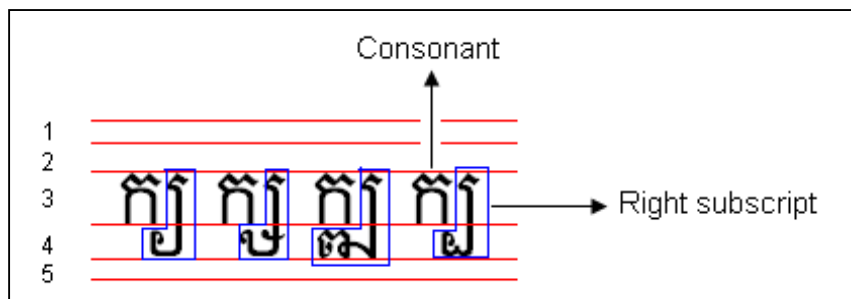


Figure 24 draw Right subscript

• drawBelowSubscript(): This is a method used to display any subtitles which render under consonant.

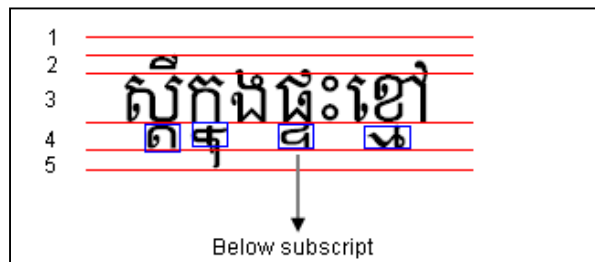


Figure 25 draw below subscript

• drawCoeungNYO(): This is a method used to display subscript of consonant NYO (ញ). Here, the subscript of NYO stays in layer 4, as in figure 26, is another shape of subscript NYO which has similar shape to its consonant. This shape occurs only if the consonant and subscript are NYO.

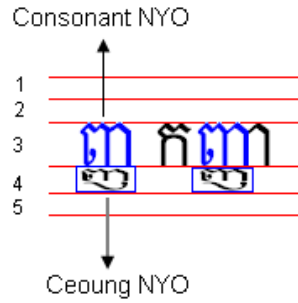


Figure 26 draw subscript of NYO

•drawBelowVowel(): It is used to draw any vowels which have rendering position below consonant. As in figure 25, there are two possible cases to display below vowel due to its rendering.

Case #1: The below vowel stands in layer 4 when it is directly under any consonants except consonant NYO (ញ) and LA (ឡ).

Case #2: The below vowel is in layer 5 when it is under consonant NYO (ញ) or LA (ឡ). Likewise, if there is a subscript (below, left, and right) belong to the main consonant, the below vowel will stand in layer 5 as well.

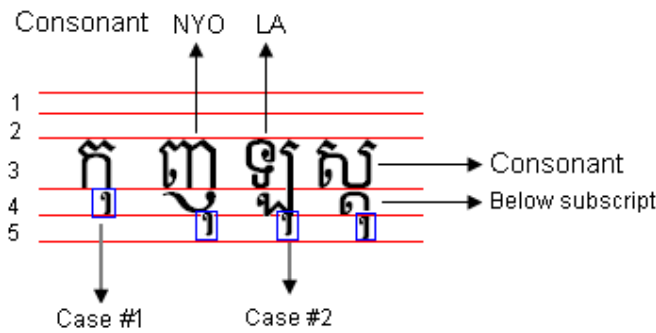


Figure 27 draw Below vowels

•drawAboveVowel(): Above vowels are displayed by using drawAboveVowel() method. As illustrated in figure 25, the method draws Above vowel in layer 1 only if there is a possible Consonant shifter (MUUSIKATOAN or TRIISAP). Otherwise the method will draw Above vowel in layer 2.

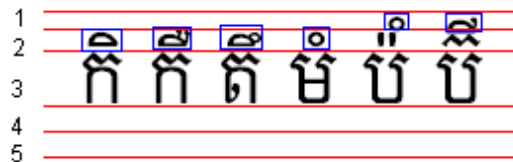


Figure 28 draw Above vowels

• drawAboveChar(): This method is similar to the drawAboveVowel() method; however, it is used to draw Khmer punctuations that render above consonant. Generally, it displays these characters in layer 2 of writing as the figure below.

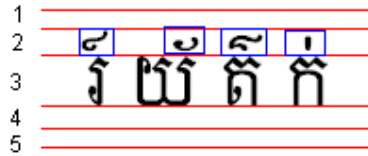


Figure 29 draw Above Characters

• drawVowelE(): It is used only to draw vowel sign E (្រៀ) that means to display itself or to complete as the left part of Two-Part vowels. See figure 27.

• drawPartVowel(): It is used only to draw the right part of Two-Part vowels. The method will be used to combine with drawVowelE() in order to display Two-Part vowel.

Figure 27 is the illustration of how to draw Two-Part vowel.

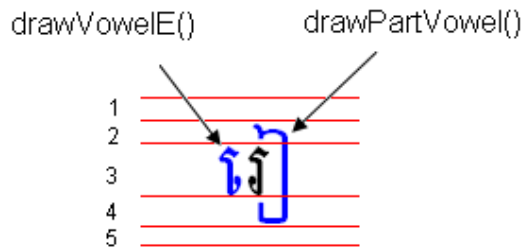


Figure 30 How to draw Two-Part Vowel

1.5.ii. Defining Khmer Character Cluster

This section describes about Khmer Character Cluster (KCC) which is a combination of characters that means inseparable unit in writing [6]. A Khmer word can be composed of a KCC or many KCCs as illustrated in the table below.

Table 2 Example of KCCs

Words	Numbers of KCCs	KCCs
ស្រ្តី	1	ស្រ្តី
ស្រ្តី	2	ស្រ្តី + ស្រ្តី
ចម្រៀង	3	ច + ម្រៀង + ង
បរិយាកាស	5	ប + រិ + យា + កា + ស

I.5.iii. Abbreviations

Here are the abbreviations used in algorithm approach which will be explained later in the next section.

Table 3 Abbreviations

C1	The 3rd series of Consonant
C2	The 1st and 2nd series of Consonant
LS	Left Subscript
RS	Right Subscript
BS	Below Subscript
CS	Consonant Shifter
LV	Left Vowel
RV	Right Vowel
BV	Below Vowel
AV	Above Vowel
2PV	Two-Part Vowel
VS	Various Sign
IV	Independent Vowel

I.5.iv. Algorithm approach

The algorithm of defining KCCs detects the possible rendering between each character of input text [6]. It bases on a transition lookup table which is provided below. In this table, current state is in rows while transition state is in columns. C1 and C2 are initial state. Starting from an initial state then continue to the next character of input string, the transition state can be possible when number 1 is met or be impossible when number 0 is met. The impossible of transition state indicates that the end of KCC is reached.

Table 4 Transition lookup table

	C1	C2	LS	BS	RS	CS	LV	RV	BV	AV	2PV	VS	IV
C1	0	0	1	1	1	0	1	1	1	1	1	1	0
C2	0	0	1	1	1	1	1	1	1	1	1	0	0
LS	0	0	0	0	0	1	1	1	1	1	1	0	0
BS	0	0	1	0	0	1	1	1	1	1	1	0	0
RS	0	0	0	0	0	1	1	1	1	1	1	0	0
CS	0	0	1	1	1	0	1	1	1	1	1	0	0
LV	0	0	0	0	0	0	0	1	0	0	0	0	0
RV	0	0	0	0	0	0	0	0	0	0	0	0	0
BV	0	0	0	0	0	0	0	1	0	1	0	0	0
AV	0	0	0	0	0	0	0	1	0	0	0	0	0
2PV	0	0	0	0	0	0	0	0	0	0	0	0	0
VS	0	0	0	0	0	0	0	0	0	0	0	0	0
IV	0	0	0	0	0	0	0	0	0	0	0	0	0

I.5.v. Rendering Rule

Prior to generating reorder of Khmer script, rendering rule needs to be defined since some Khmer characters will change their shapes to other ones. See the table provided below.

Table 5 Rendering Rule

Input String	Output String	Example
CLHV - {ឃ} + ្ល + AV - {្រ}	CLHV - {ឃ} + AV - {្រ} + ្រ	ស + ្ល + ្រ = ស្រី
CLLV + {ឃ} + ្រ + AV - {្រ}	CLLV + {ឃ} + AV - {្រ} + ្រ	ឃ + ្រ + ្រ = ឃី, ង + ្រ + ្រ = ងី
CLLV + {ឃ} + ្រ + ្រ	CLLV + {ឃ} + ្រ + ្រ	ញ + ្រ + ្រ = ញ៉
C + ្រ + ្ល + ្រ	C + ្រ + ្រ + ្រ	ផ + ្រ + ្ល + ្រ = ផ្រី

The meanings of symbols are as below:

Table 6 meaning of symbols

CLHV	Consonant Lack Heavy Voice: ង, ម, រ, យ, វ, ញ
CLLV	Consonant Lack Light Voice: ប, ស, ហ, អ
- { }	Except the character in the { }
+ { }	Include the character in the { }

I.5.vi. Reordering

Due to smart typing, type as spelling, reordering of characters must be done before displaying. After we can define KCC and rendering rule in the sections above, we start defining the order of characters in a KCC according to their types. Then, we generate reordering as illustrated in the figure below.

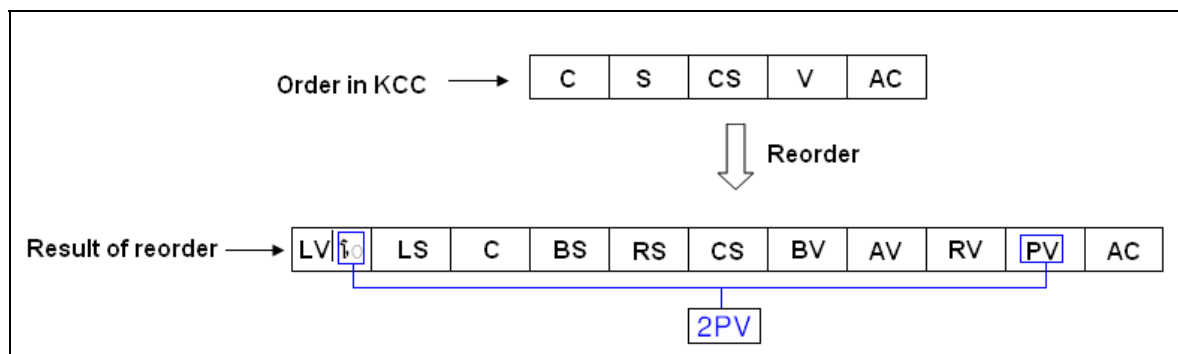


Figure 31 generating reorder in KCC

The abbreviations in the above figure are listed as the following:

Table 7: abbreviations

C	Consonant
S	Subscript
CS	Consonant Shifter
V	Vowel
AC	Above Character
LV	Left Vowel
LS	Left Subscript
BS	Below Subscript
RS	Right Subscript
CS	Consonant Shifter
BV	Below Vowel
AV	Above Vowel
RV	Right Vowel
PV	Part Vowel

The figure below is an example of generating reorder in the KCC វៀង by converting from spelling order into writing order. Finally, call each display method which appropriate for each character type.

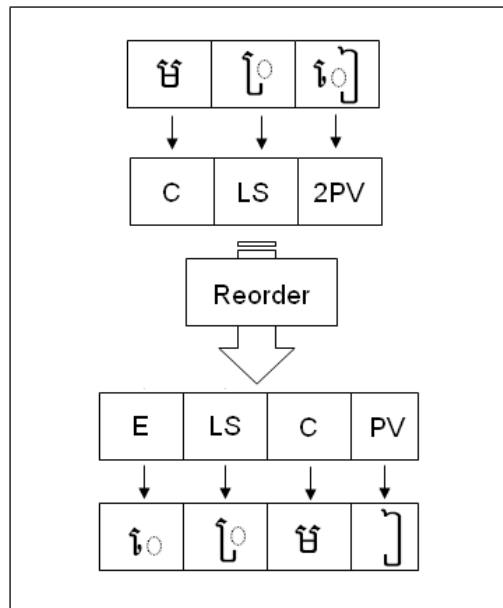


Figure 32 generating reorder in KCC វៀង

1.6. Suggested Keypad Layout for Khmer Script

Normally, standard mobile phone has 12 keys in its keypad while Khmer script has huge number of characters up to 172. Due to this reason, the keypad layout for Khmer script should be carefully taken into account. The suggested keypad layout of Khmer script is provided as in figure 30 which is designed following some suggestions:

- Commonly used characters of Khmer script should be located within key 1 to key 9. With these, we decide to assign at most 10 characters to each key due to Khmer package of five numbers (5, 10, 15, 20,etc). For instance, Key 2 (in figure 30) contains 10 characters: ក, ខ, គ, ឃ, ង, ច, ឆ, ជ, ល, and ញ.
- Consonants should be assigned within keys prior to the assignment for vowels. The reason is a Khmer word always begins with a consonant when it is written using smart typing—type as spelling. However, we initially place consonants from Key 2 while Key 1 is chosen to put only Khmer commonly used punctuations.
- After consonants, we start assigning Khmer Independent vowels, and then dependent vowels till complete all 9 keys.
- Consonants, Independent vowels, and vowels should be assigned by keeping their order as in Khmer script. This may help user to remember them correctly.
- The key * should be used to input symbols of rarely used characters such as Khmer symbols, various sign, Khmer numeric symbols of divination lore and some international symbols.
- The key 0 should be used to input COEUNG sign for Khmer script and also to input space. Thus, it contains only two symbols.
- The key # should be used for shifting languages: Khmer and English. Once it is pressed, Script will change, for example, from Khmer characters to Khmer number, from Khmer number to English number, from English number to English small letter, and from English small letter to English capital letter.

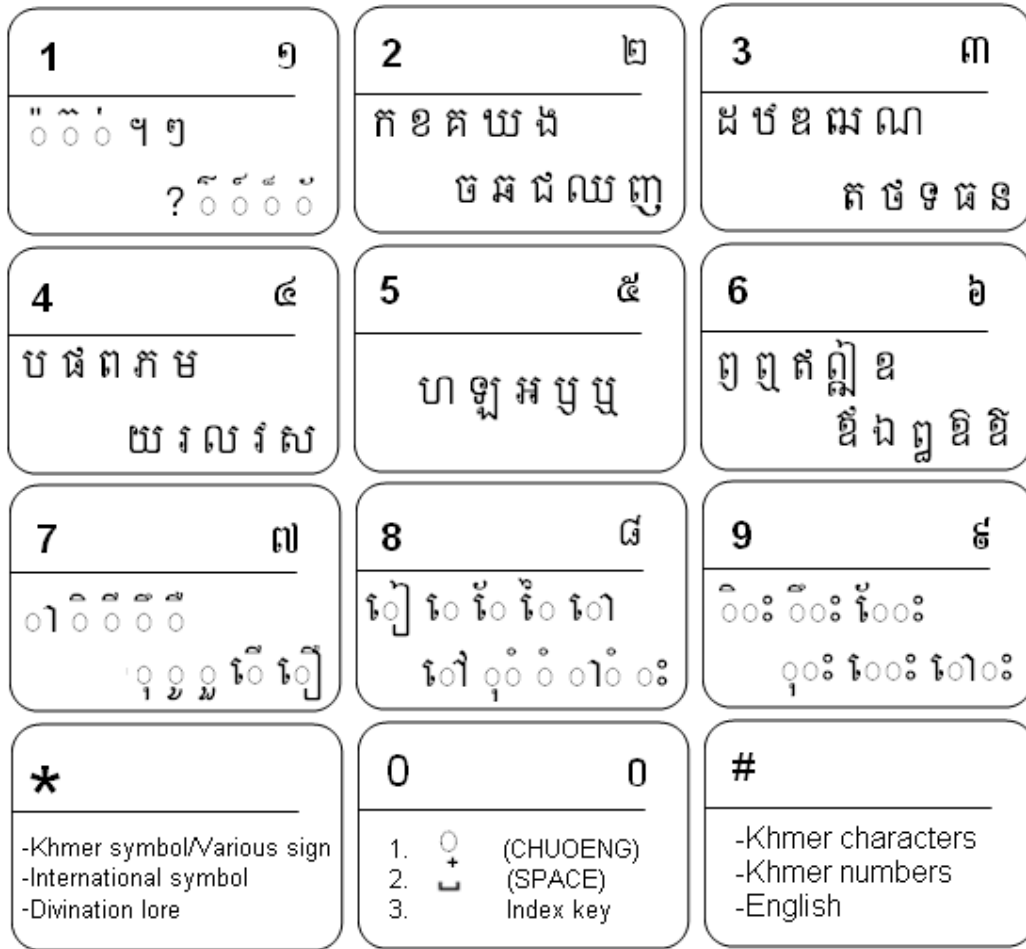


Figure 33 Suggested Keypad Layout for Khmer Script

1.7. Input Method using Two-Key press

As mentioned earlier, a key in the keypad, except key *, may contain up to 10 characters. It may cause difficulty to input Khmer text if we use multi-press, which is always seen with English text input, as the input method for Khmer script. Respond to this problem, another input method called two-key press is suggested. Two-key press means that user needs to press two keys successively in order to specify a character. First, user presses one time on a key which contains group of characters. KSMS application will popup those characters along with their index position for user to choose. Then, user can press one more key whose number equal to the index of intended character to tell the system to input that character. Once the first key is press but the user wants to change it, he/she also have option to cancel it by using cancel soft key in KSMS.

1.8. Interface Development

This section describes the interface development which makes KSMS be Khmer-base mobile application. KSMS' interfaces are created to provide users with ability of displaying Khmer script and rendering the script correctly. Furthermore, they enable users to read and write text messages on mobile phones using Khmer Unicode. Likewise, they let users to send and receive, which will be mentioned later, text messages through message service of mobile devices.

The figure provided below is the structure of Interfaces in KSMS.

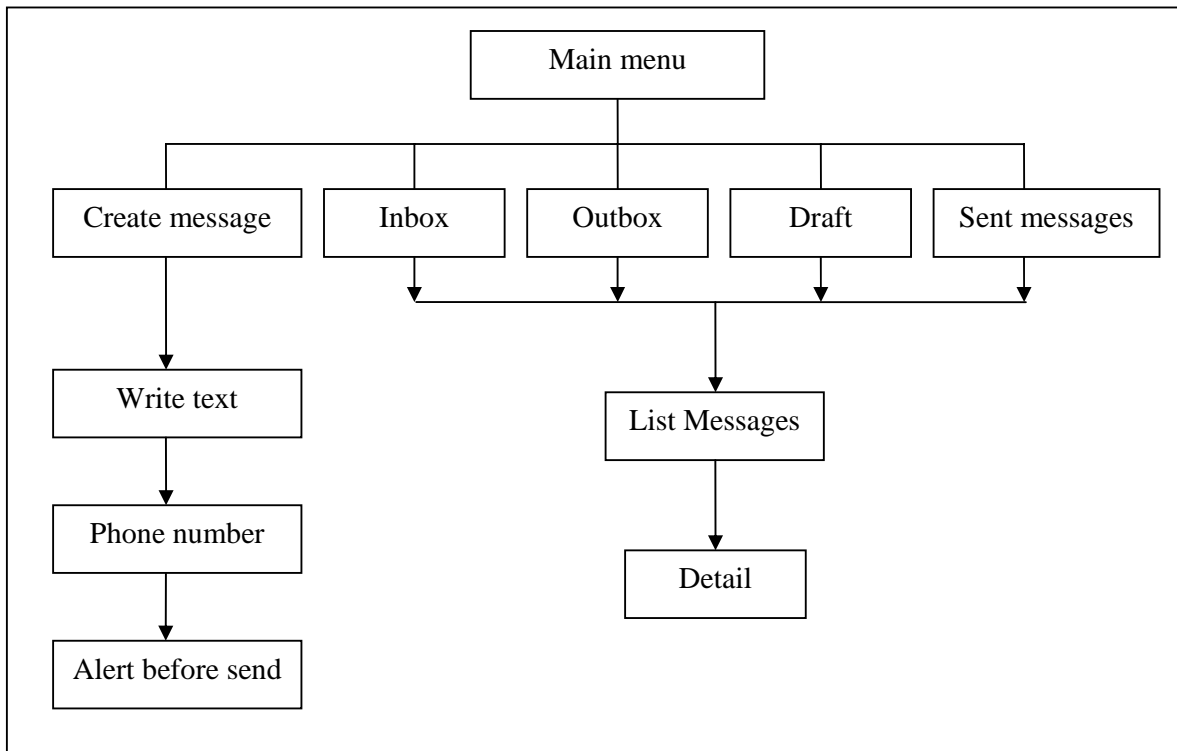


Figure 34 Structure of KSMS' interfaces

Initially, Main menu (see figure 32) is the first interface which will be shown when the application starts. It is created as a list contains fives items: Create message, Inbox, Outbox, Draft, and Sent messages. These items are displayed in Khmer script using bitmap font.

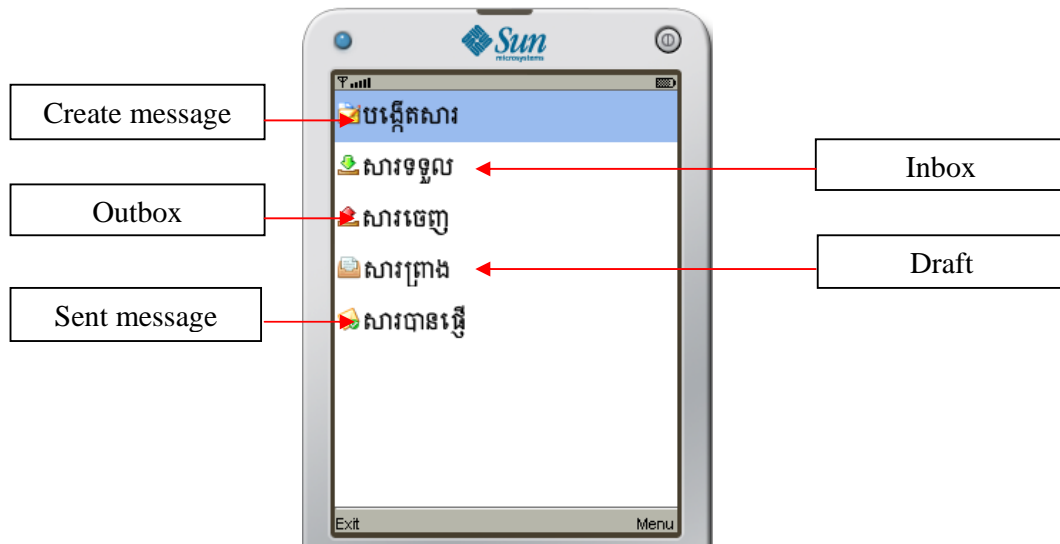
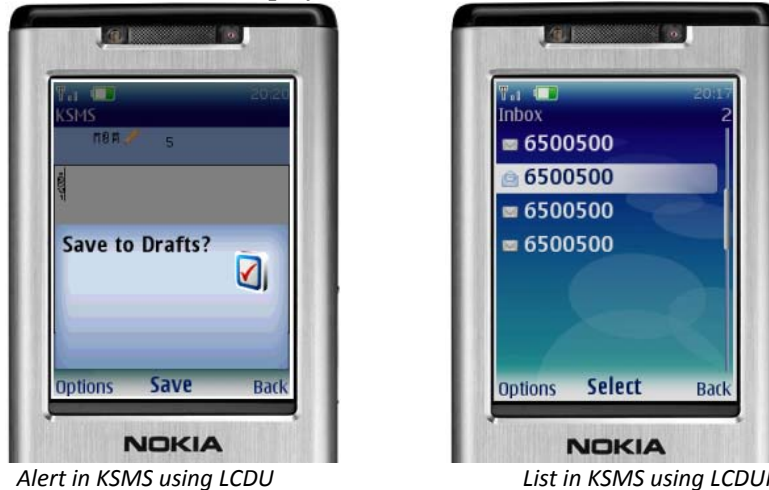


Figure 35 Main menu in KSMS

Two types of GUIs—LCDUI and microEWT—are used in order to develop such interfaces.

I.8.i. LCDUI

LCDUI (Limited Connected Device User Interface) is the base of Graphical User Interface (GUI) in J2ME (Java 2 Platform Micro Edition) [7]. Some interfaces, where Khmer script are not used, are created using LCDUI which aims to keep the theme and some built-in features of mobile devices. It means that components such as TextField, List, Alert in LCDUI will be displayed base on the appearance of mobile devices. For instance, figure 33 is the view of Alert and List of LCDUI displayed in Nokia S40.



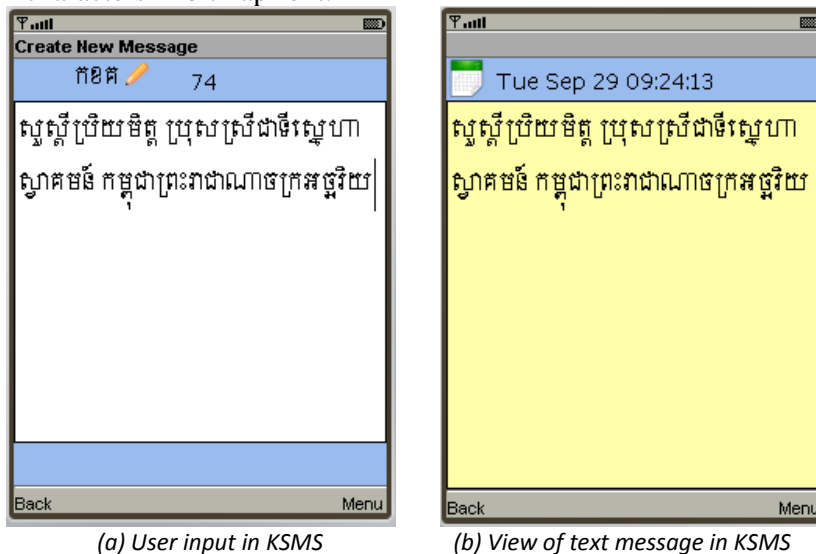
Alert in KSMS using LCDUI

List in KSMS using LCDUI

Figure 36 Alert and List using LCDUI displayed in Nokia S40I

I.8.ii. microEWT

As mentioned before, microEWT gives KSMS the ability of supporting Khmer script by rendering Khmer text using our own custom font. That means components such as TextField, Label, List, Button, Canvas, etc, in microEWT can render their text which is extracted from characters in bitmap font.



(a) User input in KSMS

(b) View of text message in KSMS

Figure 37 View of text message in KSMS

In KSMS, we use TextField of microEWT to be the place for user input and viewing text messages (see figure 34 and figure 35). The input of Khmer text must follow the smart typing that means typing bases on spelling. Rendering Engine will work to render the text before displaying it on screen of mobile phones.

After using bitmap font, List of microEWT is also able to render Khmer script in its list items. As a result, we use it to be the list of draft messages which is able to show some contents in each list item in either Latin script or Khmer script. The figure below is the view of list of draft messages in KSMS. These list items are ordered by date and time which means the newest draft is on the most top of the list while the oldest is on the lowest of the list.

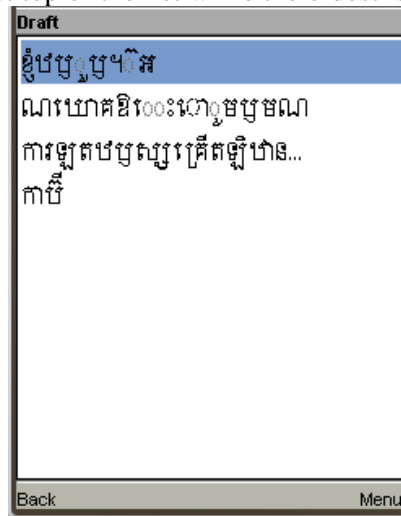


Figure 38 List of draft messages in KSMS

1.9. Send and Receive text messages

Send and Receive text messages are the features which makes KSMS be a wireless messaging application for mobile devices. To provide these features to KSMS, we use an API (Application Programming Interface) called Wireless Messaging API 1.1 (WMA 1.1) and a mechanism—Push registry.

1.9.i. Using WMA 1.1

Wireless Messaging API 1.1 is an optional package of j2me defined in JSR 120 at JCP (Java Community Process) [8]. It enables j2me application to have ability of sending and receiving short messages through wireless connection. This ability is based on Generic Connection Framework (GCF) [9], figure 36, defined in Connected Limited Device Configuration (CLDC) 1.0. WMA consists of five components which are interfaces: MessageConnection, MessageListener, Message, TextMessage, and BinaryMessage (see figure 37).

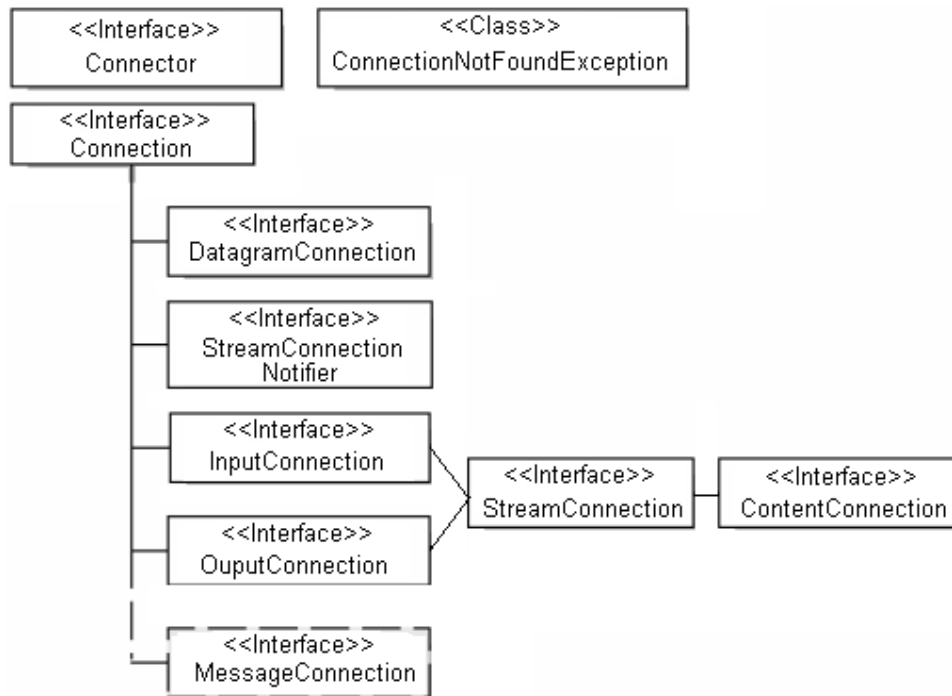


Figure 39 Generic Connection Framework (GCF)

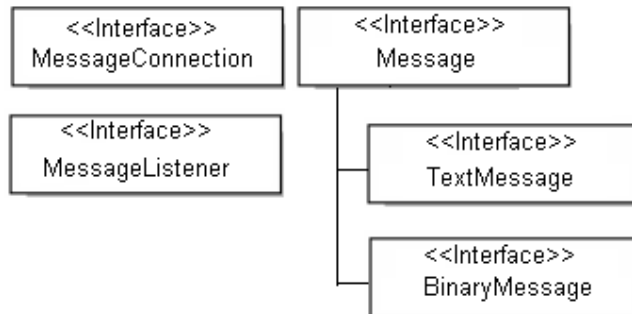


Figure 40 Components in WMA 1.1

The table below describes the components of WMA. These components reside in two separate java package such as java.microedition.io and javax.microedition.messaging.

Table 8: WMA components description

Java Package	WMA Class	Description
javax.microedition.io	MessageConnection	Factory of Message Objects. It contains methods to send receive messages. Sub interface of the GCF connection class.
javax.microedition.messaging	MessageListener	It is used to implement asynchronous notification of Message objects.
	Message	Base Message interface which Provides methods to get and set the message address, and get timestamp.

	TextMessage	It provides methods to set and get the text payload. Sub interface of Message.
	BinaryMessage	It provides methods to set and get the binary payload. Sub interface of Message

a. Port number

In order to send and receive messages, WMA involves in using a port number which ranges from 0 to 65535. Port numbers use 16 bit addressing space and can be divided into three ranges as the following:

- Range 1: 0 – 15999, allocated by the Internet Assigned Numbers Authority (IANA) (see <http://www.IANA.com>)
- Range 2: 16000 –16999, available for used by applications
- Range 3: 17000 –65535, reserved

An application (receiver) can receive messages from another application (sender) unless they both use the same port number in their MessageConnection object which will be explained later. To do so we choose 16500 (in range 2) as the port number for KSMS application using in its message connection.

b. Create MessageConnection

Functionality of sending and receiving message is implemented by a MessageConnection interface, in WMA, which is a sub-interface of Connection interface in Generic Connection Framework. MessageConnection object is obtained by using Connector.open() method with a URL connection string that identifies the address. The URL uses the format as **sms://:port number**, e.g. String URL = “sms://:5000”. The following code snippet is an example of creating message connection in KSMS using 16500 as its port number.

```

MessageConnection mc;
try {
String url = "sms://:16500";
mc          = (MessageConnection) Connector.open(url);
} catch(Exception e) {
// Handle Exception...
}
    
```

c. Sending TextMessage

Before sending, a TextMessage needs to be created by using a newMessage method of MessageConnection interface and specifying TEXT_MESSAGE. The code snippet below explains a boolean method which is used to send short text message in KSMS application; it returns true if the message successfully sent, otherwise it return false. The method is named sendSMS which requires two parameters: 1) sNumber is a String contains phone number to be sent to; and 2) sMessage is a String contains short text message.

```

public boolean sendSMS(String sNumber, String sMessage){
    boolean result = true;
    String sAddress = "sms://" + sNumber + ":16500" ;
    try {
        TextMessage msg = (TextMessage)
            conn.newMessage(MessageConnection.TEXT_MESSAGE);
        // set Address
        msg.setAddress(sAddress);
        //set text
        msg.setPayloadText(sMessage);
        // send sMessage
        conn.send(msg);
    } catch (SecurityException e) {
        result = false;
    } catch (Exception e) {
        result = false;
    }
}
return result;

```

d. Receiving TextMessage

The receiving of short text message is done by the implementation of a method named `notifyIncomingMessage`, see the code snippet below, which belongs to the `MessageListener` interface.

```

public void notifyIncomingMessage(MessageConnection
mscon)
{
    if (con == mscon)
    {
        try
        {
            Message receivedMessage = mscon.receive();
            if (receivedMessage instanceof TextMessage)
            {
                messageReceivedHandler(receivedMessage);
            }
        } catch (Exception ex) {
        }
    }
}

```

The code snippet above functions to notify when there is a message coming to our application. An object of class `Message` has been created to get data of the message such as timestamp, sender address, and content of text message. Below is the definition of method `messageReceiveHandler()` which existed in the previous snippet.

```

public void messageReceivedHandler (Message
receivedMessage)
{
    Date Timestamp =
receivedMessage.getTimestamp();
    String sSenderAddress = receivedMessage.getAddress();
    TextMessage tmTextMessage = (TextMessage)

```

I.9.ii. Using Push Registry

Push Registry is known as a mechanism used to enable j2me application to start automatically. It registers MIDlets, j2me application, to be activated whenever it detects incoming connections from a particular application [10].

KSMS application uses Push Registry in order to get the feature of automatic activation which is capable of receiving short text messages even the application is not launched yet. It means that Push Registry provides a way for KSMS to responds to an inbound connection whether it is running or not. If KSMS is not started yet, KSMS will be opened automatically and then receiving the incoming messages.

I.10. SMS Storage

This section describes how KSMS application stores and maintains its own text messages into mobile phones. Text messages—which are created, sent, and received by KSMS application—are categorized according to their types such as inbox messages, sent messages, draft messages, and outbox messages. The way of storing information of messages into mobile phones is done by using a mechanism called Record Management System (RMS), a persistent storage for MIDlet.

I.10.i. Understanding Record Management System (RMS)

Record Management System (RMS) is a data persistence mechanism provided by MIDP. In KSMS, RMS has been used to store information of text messages (timestamp, sender address, and content of message) in terms of records in record store.

Record is an individual data item stored in a record store [11], which will be address later. Each data item can be any types: number, string, image, array, or anything that can be presented by a sequence of byte called byte array. More than this, a record always has its own ID, an integer started from 1, which is automatically assigned by RMS.

Record Store is an ordered collection of records [11]. It is identified by a name, case-sensitive and not more than 32 characters long. It is noted that an application cannot own two records with the same name.

I.10.ii. Why RMS?

Mobile phones have less capacity which limited their behavior in the manner different from big devices like powerful devices such as PC, server, etc. They do not have a robust file, therefore, cannot store data in the way same to any powerful devices. However, RMS (Record Management System) helps mobile phones to have ability to store information effectively and efficiency. It means that RMS provides a file system—like environment to store and maintain persistent data [12]. In addition, each record in record stores of RMS can be managed: insert, read, write, sort, and search easily.

I.10.iii. The use of RMS in KSMS

Record stores of RMS are used to be directories for storing each type of text messages in KSMS. According to KSMS has four types of text messages, i.e. inbox messages, outbox messages, draft messages, and sent messages, four directories need to be created for using in KSMS. That means four record stores need to be created as well. In that, a text message is represented by a record which exists in its record store.

a. Create record store

A record store can be created by using `RecordStore.openRecordStore()` method with the second parameter set to true as demonstrated in the code snippet below.

```
// Create record store
String sNameOfRecordStore = "khINBOX"
RecordStore rs = null;
Try
{
    Rs = RecordStore.openRecordStore(sNameOfRecordStore,
true);
} catch (RecordStoreException e){
    // couldn't create or open it
```

b. Adding and Updating Records

To add a record to record store, `RecordStore.addRecord()` method has been used. The record must be array of byte. See the code snippet below.

```
...
byte[] data = new byte[]{ 0, 1, 2, 3 };
int recordID;

recordID = rs.addRecord( data, 0, data.length );
...
```

A record also can be updated by using another method which is `RecordStore.setRecord()` as shown in the snippet below.

```
...
int recordID = ...; // some record ID
byte[] data = new byte[]{ 0, 10, 20, 30 };

rs.setRecord( recordID, data, 1, 2 );
    // replaces all data in record with 10, 20
```

c. Reading Records

Reading records means retrieving it from its record store. To do so we use `RecordStore.getRecord()` in which an integer is needed as its parameter to identify the record ID (see the code snippet below).

```

...
int    recordID = .... // some record ID
byte[] data = rs.getRecord( recordID );
...

```

d. Delete records and Record Stores

Records or record stores can be deleted. A record can be deleted through the method `RecordStore.deleteRecord()` and using a parameter, an integer, to identify the specific record to be deleted. The code snippet provided here tells how to delete a record with a record ID.

```

...
int    recordID = .... // some record ID
rs.deleteRecord( recordID );
...

```

Once a record is deleted, any attempt to use it will cause `InvalidRecordIDException` to be thrown.

Likewise, a record store can be deleted by using `RecordStore.deleteRecordStore()` with a string parameter specify the name of record store. However, it can be deleted only if it is not currently open, and only by a MIDlet in the owning MIDlet suit. As provided below, it is the code snippet tells how to delete a record store.

```

...
try {
    RecordStore.deleteRecordStore( "myrs" );
} catch( RecordStoreNotFoundException e ){
    // no such record store
} catch( RecordStoreException e ){
    // somebody has it open
}

```

II. Experiment Result

In our experiments, we tested the rendering performance in KSMS application with many words extracted from the Chhuon Nath dictionary [13]. With an emulator in computer, the result is that rendering engine works fine with most cases of Khmer words. Some cases cannot be rendered correctly due to wrong order of typing characters in those words. For example, the word ស្រ្តី can be typed in two ways: (1) ស + ្រ + ្តី and (2) ស + ្រ + ្តី + ្រ + ្តី. The first order is generally accepted by Khmer grammar rule while the second one, the alternative case of the first, used by some Cambodian users. However, it can be said that the second order is the miss spelling of the word ស្រ្តី. As a result, the word ស្រ្តី is rendered incorrectly in KSMS whenever it is typed using the second sequence of characters.

In addition to the rendering performance, we tested the speed of KSMS in real mobile phones. Here we choose two mobile phones: Nokia 6300 and Sony Ericsson W595 which are tested by three people. The result we are going to mention below (table) is focus on time consuming during the typing process of each sentence. Time is shown in format mn:ss:ms in which mn = minutes, ss = second, and ms = millisecond.

Table 9 Time consuming during typing process in KSMS

N	Sentence	Person A		Person B		Person C	
		6300	W595	6300	W595	6300	W595
1	តើអ្នកសុខសប្បាយជាទេ?	1:20:46	2:33:67	1:40:51	2:35:36	0:44:97	1:00:47
2	ខ្ញុំនឹកអ្នកខ្លាំងណាស់	0:52:43	1:49:42	0:44:56	1:15:58	0:58:01	1:15:01
3	នារីពុំប្រកែកនឹងកញ្ញាទេ	1:01:14	2:27:43	1:27:24	2:50:08	0:52:78	1:03:83
4	មូលដ្ឋានីយកម្មអក្ខរក្រមខ្មែរ	1:00:04	2:19:81	1:42:23	3:03:85	1:09:38	1:36:32
5	ម្តាយធីតាទៅផ្សារ	0:43:17	1:28:11	1:17:61	2:25:05	0:42:08	1:11:57
6	កូនក្មេងយំនៅសួនច្បារ	0:44:40	1:43:26	2:35:36	2:09:92	0:53:45	1:24:16
7	ចិន្តាបង្រៀនកូនសិស្ស	0:49:57	1:39:53	1:15:58	1:12:25	0:43:40	1:19:50
8	ពេលប្រជុំត្រូវស្ងៀមស្ងាត់	1:02:69	2:16:03	2:50:08	2:36:92	0:55:76	1:19:50
9	ប្រយ័ត្នផ្នែកកាត់ក្បែររបង	1:02:57	2:07:98	3:03:85	2:30:09	0:55:07	1:13:08
10	បរិស្ថានកាលឆ្នាំសិក្សា ២០០៩-២០១០	1:02:66	1:35:69	2:25:05	2:05:95	1:08:98	1:30:86

III. Conclusion

KSMS application is a Khmer-base application created to provide Cambodian people to use SMS in Khmer script for their communication. Since the rendering engine in KSMS is created base on the rule of Khmer grammar, KSMS has ability to display Khmer script correctly for most cases of Khmer words. Besides rendering, KSMS can send and receive short text messages through wireless network. Furthermore, all text messages using in KSMS are stored in proper directories created in KSMS: Inbox, Outbox, Draft, and Sent message. The performance of the application may be different from phone to phone that means some models of mobile phone can operate KSMS smoothly while the others cannot. However, the application still needs more improvement and modification to make it be compatible to most of mobile devices.

References

- [1] K Thong Huot Telecom Co.,Ltd., "Khmer phone products," K Thong Huot Telecom Co.,Ltd., 2009. [Online]. Available: <http://www.kth.com.kh/products.php?x=5&xx=0&active=productM> [Accessed: Sept. 07, 2009]
- [2] "ផ្ញើសារអេសអឹមអេសជាខ្មែរឃ្លានិក្ខណ៍," 2008. [Online]. Available: <http://www.khmerthmey.com> [Accessed: Mar. 06, 2009]
- [3] Forum Nokia, "Forum Nokia - Mobile J2ME Applications," 2009. [Online]. Available: http://www.forum.nokia.com/Technology_Topics/Development_Platforms/Java.xhtml [Accessed: Sept. 07, 2009]
- [4] SUN Developer Network (SDN), "Java ME - the Most Ubiquitous Application Platform for Mobile Devices," 2009. [Online]. Available:[Accessed: Sept. 07, 2009]
- [5] Esoco GmbH, "microEWT," esco.net, 2007. [Online]. Available: <http://www.esoco.net/content/view/7/1/>. [Accessed: Jul. 28, 2009].
- [6] R. Pich Hemy and V. Navy, "Khmer Word Segmentation Project Report," PAN Localization Cambodia (PLC) of IDRC, 2006, pp. 22-24

- [7] Nokia, *Java Developer's Library 3.3*. Nokia Forum: Nokia, 2009. [E-book] Available: Nokia Library
- [8] Sony Ericsson. *Advanced Message Coding Using JSR 120*. Sony Ericsson Developer World: Sony Ericsson, November 2003. [E-book] Available: Sony Ericsson Developer World.
- [9] C. Enrique Ortiz, "The Generic Connection Framework," Sun Developer Network (SDN), August 2003. [Online], Available: <http://developers.sun.com/mobility/midp/articles/genericframework/> [Accessed: Sep. 21, 2009]
- [10] Enrique Ortiz, "MIDP 2.0 Push Registry," Sun Developer Network(SDN), January 2003. [Online]. Available: <http://developers.sun.com/mobility/midp/articles/pushreg/>. [Accessed: Jul. 28, 2009]
- [11] Eric Giguere, "Databases and MIDP, Part 1: Understanding the Record Management System," Sun Developer Network (SDN), February 2004. [Online]. Available: <http://developers.sun.com/mobility/midp/articles/databaserms/> [Accessed: Jul. 29, 2009]
- [12] Sina koca, "NetBeans Wiki: RMS Tutorial," NetBeans.org, 19 November, 2008. [Online]. Available: <http://wiki.netbeans.org/RMSTutorial>. [Accessed: Jul. 28, 2009].
- [13] Chuon Nath, *Cambodian dictionary*, Edition of Buddhist Institute, Phnom Penh, 1967