Enabling Complex Asian Scripts on Mobile Devices

Waqar Ahmad
Computer Science Department,
National University of Computer and Emerging Sciences, Lahore, Pakistan
waqar.ahmad@nu.edu.pk

Sarmad Hussain
Center for Language Engineering,
Al-Khawarzimi Institute of Computer Science, University of Engineering and Technology, Lahore, Pakistan
sarmad@cantab.net

Abstract—*The increasing penetration of mobile devices has resulted in their use in diverse domains such as education, health, entertainment, business, sports, and social networks. However, lack of appropriate support for many local languages on mobile devices, which use complex scripts rather than Latin scripts, is constraining many people across developing Asia and elsewhere from using their mobile devices in the same way. There are some ad hoc solutions for certain scripts, but what is needed is a comprehensive and scalable framework which would support all scripts. The Open Type Font (OTF) framework is now being widely used for supporting complex writing systems on computing platforms. If support for OTF is also enabled on mobile devices, it would allow them to also support complex scripts. The current paper reports on work in this area, taking Pango, an open source rendering engine, and porting and testing its language modules on a mobile platform to provide support for Open Type Fonts. The paper describes the process for successful deployment of this engine on Nokia devices running the Symbian operating system for Urdu, Hindi and Khmer languages. The testing results show that this is a viable solution for enabling complex scripts on mobile devices, which can have significant socio-economic impact, especially for developing countries.*

Keywords— *Mobile Devices, Smart-Phones, Pango, Localisation, Open Type Fonts, Complex Writing Systems*

I. Introduction

The number of mobile phone subscriptions worldwide is expected to reach 5 billion in 2010 (ITU 2010). Mobile device penetration in developing countries of Asia is also increasing at a rapid pace (MobiThinking 2010). While current and past usage of mobile devices has mostly been for voice, there is a significant increase in text and other data services using smart-phones (adMob 2010). It is expected that more than 85% of mobile handsets will be equipped for mobile web access by the end of 2011 (MobiThinking 2010), as many smart-phones today have processing power and other capabilities comparable to desktop computers of early 1990s.

As the hardware capabilities of mobile devices improve, they are increasingly being used in areas like education, health, entertainment, news, sports, and social networks. This usage of smart-phones requires that text and other data services are made available in local languages. However, most of the mobile devices that are currently in use only support Latin script. There is limited or no support available for many other language scripts, specifically those of developing Asia. The devices generally support basic Latin, bitmap and True Type Fonts (TTF). Most Asian languages scripts, on the other hand, are very cursive, context sensitive and complex (Hussain 2003; Wali and Hussain 2006), and can only be realized using more elaborate font frameworks, e.g. Open Type Fonts (OTF) (Microsoft 2009). Such frameworks are not supported on most mobile devices and smart-phones at this time. Many people in developing Asia are only literate in their own languages and are, therefore, unable to utilize their mobile devices for anything other than voice calls. Developing font support is an essential pre-cursor to making content available in local language scripts. Once support is in place, content can be created, allowing people to utilize the additional capabilities of mobile phones for their socio-economic gains.

Whether focusing on iPhone (Apple Inc. 2010), Symbian based Nokia Phones (Forum.Nokia Users 2009), Google Android (Google 2009), Windows Mobile (Microsoft 2010), or Blackberry, the worldwide web is full of queries and posts showcasing the needs and concerns of developers and end-users, who are looking for particular language support on their devices. While there is extensive localisation support for desktop computers mobile devices are lagging behind. Smart-phone software developers try to find workarounds for resolving localisation issues and sometimes achieve limited success. However, total success can only be achieved if the underlying device platform provides comprehensive support. If the underlying platform has limitations, then they are also reflected in the workarounds produced by software developers. A major problem is that mobile platforms provide limited software internationalisation support and therefore, localisation for certain languages may become very difficult.

In this paper we have suggested a solution for alleviating some of the problems associated with the support of complex Asian scripts on mobile devices using Pango—an open source library for text layout and rendering with an emphasis on internationalisation (Taylor 2004). Research and development has been carried out with a focus on evaluating the viability of Pango as a text layout and rendering engine on mobile platforms. For this project, Symbian has been chosen as the mobile platform. The project has two components: one component deals with porting script specific modules of Pango to the Symbian platform; the other component is the development of an application (referred to as the SMSLocalized Application hereinafter) that can send/receive SMS in local languages using Pango with mobiles, as a proof of concept.

Although all of the language specific modules of Pango are ported successfully to Symbian platform, extensive testing is performed for Urdu and an initial level of testing is performed for Khmer and Hindi. The results of the tests are quite promising and confirm the viability of Pango as a font engine for mobile devices. The SMSLocalized application contains features specialized for local language scripts. This application has been tested for Urdu; however, the architecture of the application is very flexible and as such allows quick application customization for other languages. This paper presents the relevant background and details of this work.

II. Current Localisation Support on Mobile Platforms

Limitations in script support on mobile devices are often due to constraints specific to mobile handsets such as a small amount of memory, limited processing power and other factors. During our research, we have learnt that most of the issues related to localisation on mobile phones fall in one or more of following patterns:

- Localisation features supported on a mobile device may not be adequately documented. As a result of this, information about localisation features may only become known after acquiring and evaluating the device by installing localised software.
- Only a limited set of features for a language may be supported on the device. For instance, True Type Fonts (TTF) may be supported but not Open Type Fonts (OTF), which will results in lack of support of a various languages and their scripts.
- In mobile device system software, language support may exist at the level of menu items but may be missing at application level. For instance, a device may have an operating system with a properly localised user interface but an on-device messenger application may not allow the user to input text in a local language.
- A particular device platform may support many languages as a whole. However, when a device is released in the market, it may only be equipped with a subset of the platform's supported languages. For instance, a language-pack may be missing or the font rendering engine may be constrained by its multilingual language support.

As previously mentioned, software developers continue trying to find workarounds for the localisation issues which are, in many ways, limited by the support provided by the underlying device platform. The following sections give an overview of the extent of localisation support on some of the major smart-phone platforms.

A. *Symbian*

Symbian OS, currently owned by Nokia, is the most widely deployed operating system on mobile phones. It supports application development using Java Micro Edition (Java ME) and C/C++. Symbian operating system supports a very basic level of user interface which does not make it usable by layman users. Therefore, on top of Symbian operating system, some mobile device vendors have developed rich user interfaces. Two such user interfaces are S60, developed by Nokia, and UIQ, developed by UIQ technology. (Morris 2007).

Symbian supports a number of languages. However, it does not support Open Type Fonts (Forum.Nokia 2009). Its default engine is based on the FreeType font library (Forum.Nokia 2009). The Symbian operating system, however, can be extended by plugging in an external font engine to add support for languages or scripts not already supported (Morris 2007). For instance, an engine can be developed or adapted from open source that adds support for open type fonts with complex scripts i.e. if a third party developer wants open type font support, s/he can develop and plug the font engine into the operating system which can then be used by any software application on the device.

B. *Windows Mobile*

Windows Mobile is a Windows CE based operating system developed by Microsoft. Windows CE is primarily designed for constrained devices like PDAs and can be customized to match the hardware components of the underlying device (Microsoft 2010). Windows Mobile supports the Microsoft .Net Compact Framework for application development, which in turn supports a subset of Microsoft .Net Framework features.

According to the Microsoft website (Microsoft 2010), WordPad, Inbox, Windows Messenger, and File Viewer applications are not enabled for complex scripts like Arabic, Thai, and Hindi.

There are some commercial solutions for localisation on the Windows Mobile platform. One such solution is Language Extender. It supports Arabic, Czech, English, Estonian, Farsi, Greek, Hebrew, Hungarian, Latvian, Lithuanian, Polish, Romanian, Russian, Slovak, and Turkish (ParaGon Software Group 2010). However, Open Type Fonts for other complex writing systems, e.g. Urdu Nataleeq (Wali and Hussain 2006) are not available.

C. *Android*

Android is a relatively new mobile software stack based on Linux. It allows application development using the Java programming language. However, a native SDK is also available from the Android developer website that can be used to develop native applications in C/C++ (Google 2010).

Localisation on the Android platform is still limited to a few languages. Independent developers have tried workarounds with limited success (Kblog 2009). There is lot of debate on language support issues on Android forums (Google Android Community 2010). However, it has still not been made clear, officially, from Google as to when support for OTF will be included.

Google (2009) talks about localisation for German, French, and English but does comment about languages using non-Latin scripts.

D. Apple iPhone

According to Apple (Apple 2010), the Apple iPhone 3G supports a number of languages including English (U.S), English (UK), French (France), German, Traditional Chinese, Simplified Chinese, Dutch, Turkish, Ukrainian, Arabic, Thai, Czech, Greek, Hebrew, Indonesian, Malay, Romanian, Slovak, and Croatian. Again, only TTF based fonts, e.g. for Arabic script, are supported, and OTF fonts are not supported.

E. Monotype Imaging Rasterization and Layout Engines for Mobile Phones

Monotype imaging (2010) provides engines for font rasterization (iType Font Engine) and layout (WorldType Layout Engine) for smart-phones. The solution is ANSI C based and is available for integration with Android, Symbian and Windows CE. However, full Open Type Font support is not available in their solutions.

F. Other Smart-phone Platforms

Other smart-phone platforms like RIM Blackberry, Palm WebOS etc. are not investigated in detail from a localisation perspective in the current work. They support localisation features, however, their limitations are similar to those mentioned above, as are discussed on online developer and end-user forums (ParaGon Software Group 2010).

III. CURRENT WORK

An investigation is conducted to evaluate the possibility of using Pango as a text rendering and layout engine for smart-phones. The project covers the following:

1. Porting language specific modules of Pango to the Symbian operating System.
2. Development of an SMS application (SMSLocalized), designed so that it can be customized for scripts supported by Pango.

As Symbian is a dominant and mature mobile platform, it has been chosen for this project. Pango has a basic module and multiple scripts specific modules, e.g. for Arabic/Urdu, Indic, Khmer, Tibetan, etc. There has already been a compilation of Pango for the Symbian platform (Cairo Graphics 2009), however, this compilation only covers the basic module, and script-specific modules have not been ported. We use Cairo and compile individual script modules on Symbian. Among the modules ported, Arabic (for Urdu), Indic and Khmer are tested after deployment. The rest of the paper is focused on this process of porting and testing the script specific modules of Pango on the Symbian platform.

A. Symbian Overview

As said earlier, Symbian OS is currently the most widely deployed operating system on mobile phones. It supports application development using Java and C/C++. Java application development on Symbian is enabled using Java Micro Edition (Java ME) and C/C++ application development is enabled using native OS application framework. (Morris 2007). To fully exploit native device features, development in C/C++ is required. Therefore, for this project, which requires extensive native device features, the development is also carried out in C/C++. A typical Symbian C/C++ application is designed according to Model-View-Controller (MVC) architecture (Harrison and Shackman 2007). SMSLocalized Application has also been developed according to the same MVC architecture.

As Pango is a C based library (Martensen 2009), Symbian support for C/C++ makes it easier to port the library. Depending upon the type of features accessed by an application from the device operating system, a Symbian application may require official signing from Symbian Signed. For development and testing of our application, we used the 'developer certificates.'

B. Pango Overview

Pango is a popular text layout and rendering library used extensively on various desktop platforms. Pango is the core library used in GTK+-2.x for text and font handling (Martensen 2009; also Taylor 2004). Pango has a number of script specific modules, including modules for Arabic, Hebrew, Hangul, Thai, Khmer, Syriac, Tibetan, and Indic scripts. Pango can work with multiple font back-ends and rendering libraries as mentioned below (Martensen 2009).

- Client side fonts using the FreeType and Fontconfig libraries. Rendering can be done with Cairo or Xft libraries, or directly to an in-memory buffer with no additional libraries.
- Native fonts on Microsoft Windows using Uniscribe for complex-text handling. Rendering can be done via Cairo or directly using the native Win32 API.
- Native fonts on MacOS X using ATSUI for complex-text handling. Rendering using Cairo. ATSUI is the library for rendering Unicode text on Apple Mac OS X.

C. R&D Challenges

Mobile application development poses a lot of challenges primarily due to the constrained nature of the devices. Limited memory size, low processing power, dependency on batteries, constrained input and output modalities, limited system APIs access, are just some of the many constraints faced by mobile application developers and researchers.

While the support for high level application development for mobile devices is extensively available, low-level application development remains challenging. Even more challenging is exploring areas which are relatively lesser traversed by application developers and researchers e.g. localisation and font rendering. Lack of documentation, few forum discussion threads, scarcity of expert developers, the unpredictable nature of development and the limited debugging and testing platforms, are among some of the major challenges that we faced during project R&D on localisation for smart-phones. Even installation of a font file on a mobile device may at times become a challenge. For example, it is not always easy to find out where to copy font files, how to get the device to detect a new font etc. Details such as these may only be known after extensive exploration of the device platform under consideration, as it may be documented well for application developers.

D. Libraries

Integration of Pango with Cairo provides a complete solution for text handling and graphics rendering. The combination of Pango and Cairo, along with their dependencies, is compiled for the Symbian platform as part of this project. The following libraries are required for complete solution to work properly:

1) *Pango*
Pango is font rendering and text layout engine available with an open source license. Pango has a number of language specific modules, including modules for Hebrew, Arabic, Hangul, Thai, Khmer, Syriac, Tibetan, and Indic scripts (Martensen 2009), as discussed.

2) *Cairo*
Cairo is 2-D graphics library which supports multiple output devices i.e. X-Window, Win32, PDF, SVG etc. The library has been written in the C programming language; however, its bindings are available in other languages such as Java, C++, PHP etc. (Cairo Graphics 2010).

3) *FreeType*
FreeType is an ANSI C compliant font rasterization library. It provides access to font files of various formats and performs actual font rasterization. Font rasterization features include the conversion of glyph outline of characters to bitmaps. It does not provide APIs to perform features like text layout or graphics processing (Free Type 2009).

4) *FontConfig*
FontConfig allows the selection of an appropriate font given certain font characteristics. It supports font configuration and font matching features and depends on the Expat XML parser. FontConfig has two key modules: The *Configuration Module* builds an internal configuration from XML files and the *Matching Module* accepts font patterns and returns the nearest matching font (FontConfig 2009).

5) *Glib*
GLib provides the core application building blocks for libraries and applications written in C. It provides the core object system used in GNOME, the main loop implementation, and a number of utility functions for strings and common data structures (Pango 2009).

6) *Pixman*
Pixman is a low level pixel manipulation library for X and Cairo. Supported pixel manipulation features include image compositing and trapezoid (Pixman 2009).

7) *Expat*
Expat is an XML parsing library written in C. It is a stream-oriented parser in which an application registers handlers for components that the Expat parser might find in the XML document e.g. XML start tags (Expat 2009).

8) *libpng*
Libpng is a library written in C for the manipulation of images in PNG (Portable Network Graphics) format (Roelof 2009).

E. Tools and Technologies

The following tools and technologies are used for the development of this work.

1) *Code Baseline*

Code from http://code.google.com/p/cairo-for-symbian/ (Cairo Graphics 2009) is taken as baseline for the current work. This is an earlier compilation of the basic Pango module for Symbian platform.

2) *Development Tools*

The Following tools were used during development:

- Carbide C++ v2.3.0: an IDE provided by Nokia for application development on the Symbian platform (Forum.Nokia 2009).

- Symbian S60 3rd Edition Feature Pack 2 SDK v1.1.2: a development kit for Nokia S60 and Symbian platforms. It includes a simulator for testing applications on a Windows desktop before they are installed and tested on actual devices (Forum.Nokia 2009).

F. Application Architecture

The project has two major parts. The first is an SMS application for testing font support and porting of the language modules of Pango and development.

1) SMSLocalized Application

The SMSLocalized application is a Symbian application designed for the languages supported through Pango. The application has the following features.

- Allows typing of text using an SMS Text editor.
- Displays an on-screen keypad, which is configurable based on a text-file for a language.
- Sends and receives text as SMS.
- Automatically wakes up whenever a new message is received.

The SMSLocalized application is implemented for the Urdu language, chosen for its complexity in contextual shaping and positioning of glyphs (Hussain 2003).

Figure 1 depicts SMSLocalized application class diagram developed in Symbian C/C++. *SMSLocalized Application*, *SMSLocalizedDocument*, *SMSLocalizedAppUi*, and *NewMessageContainerView* are required by the MVC architecture of Symbian applications.

To enable Urdu text input on mobile phones, a custom key map has to be defined so that the appropriate Urdu characters of Urdu are rendered against each key press. Many mobile phones support multi-tapped text input, where each key on the keypad represents more than one characters. This arrangement of character sequences against each numeric key on the mobile phone is called the keymap i.e. each numeric key on the device has an associated keymap.

On a typical Symbian device, a keymap is defined against each key on the device keypad so a character can be entered using the multi-tapping nature of Numeric keypads. *NumerciLocalizedPtiEngine* provides customized low level input mechanisms. One key feature supported in this class is that it defines a new keymap for the local language. *NumericKeypad* is used to draw a custom localized keypad on the mobile screen. This involves measuring screen size and dividing it appropriately to allow sufficient space for a numeric keypad consisting of four rows and three columns while still giving enough space to enter text. The *CSMSWatcher* class inherits from *CActive* and registers an active object with the scheduler. It implements methods to handle messages received by the application.

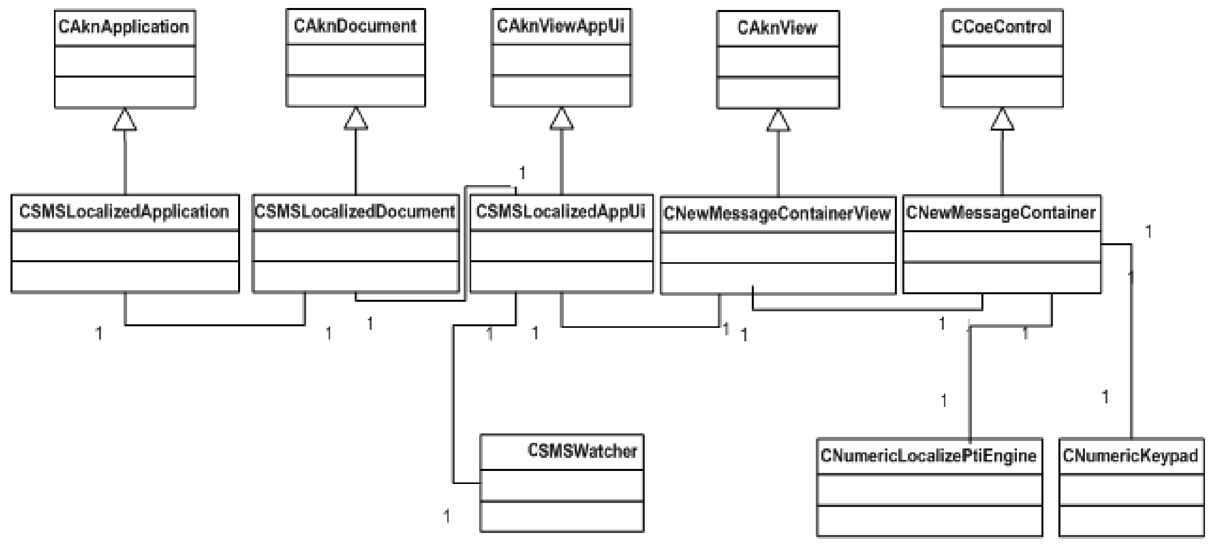

Figure 1: Class Diagram of the SMSLocalized Application

To prevent the Symbian operating system from loading the default keymap and using the customized keymap for another local language, a new keymap has to be defined and a mechanism developed to load this sequence of characters when the application starts up. This involves defining a custom Unicode sequence against each key on the numeric keypad in a text file and using the *CPtiEngine* API of the Symbian platform to load customized keymap sequences from the relevant resource file.

2) Script Specific Modules of Pango

The second major component of the solution is the Pangocairo library core and script-specific modules. The Pangocairo library, along with script-specific modules, are compiled and ported to Symbian platform.

Pango supports multiple scripts including Latin, Cyrillic, Arabic, Hangul, Hebrew, Indic and Thai. Figure 2 provides an overview of the high level architecture of Pango (Taylor 2001). The following are key features of the Pango Architecture (Taylor 2001):

- Unicode has been used as common character encoding mechanism throughout the Pango system.
- There is a core functionality module, Pango Core, which includes functions such as itemization (subdivision of text strings) and line breaking.
- There are script specific modules for handling features unique to each script. Each script module has been further split into two modules: the *language module* and the *shaper module*. The *language module* is independent of rendering system and the *shaper module* (e.g. Arabic X Shaper, PS X Shaper) is dependent on the rendering system.
- Pango rendering components support multiple rendering back ends. There are separate components for each rendering backend e.g. *X rendering backend* is responsible for rendering X fonts using XLib and XServer.

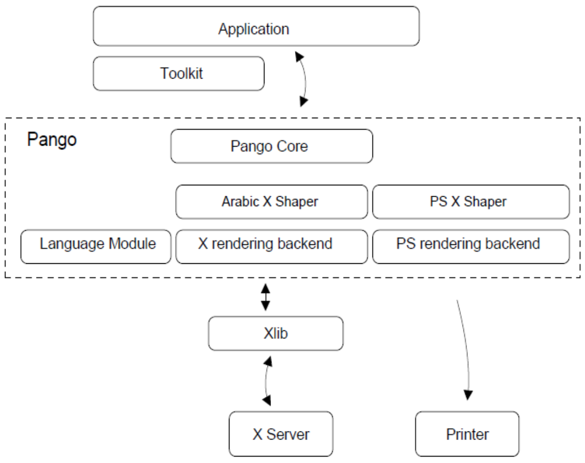

Figure 2: Pango Architecture (Taylor 2001)

Pangocairo itself includes packages of standard C/C++ libraries. Therefore, it can be ported to the Symbian platform, which also supports C/C++. However, this task is challenging because the availability of the technical information required is limited. The following are some important modifications carried out in Pango and its dependent libraries in order to port it onto the Symbian operating system.

- Declarations of language specific modules are included in the code, which lead to the generation of interface functions. These interface functions enable access to the language specific modules in the code.
- The source code that needs to be compiled for the Symbian operating system must be referred to in appropriate 'project make files' i.e. .mmp files. References to interface components of script specific modules (e.g. Arabic) are included in appropriate .mmp files.
- On start-up, the Symbian operating system loads font files from specific folders. Since the FontConfig library accesses font files, it is updated so that it can access Nafees Nastalique font files loaded by the Symbian operating system.
- Some of required Pango API functions are not exposed for external access in the Symbian code. Such functions are declared and listed in appropriate interface files.

In addition to the above, a component that interfaces with the Pango library has been created. This component enables access to text rendering features of Pango i.e., it can take any Unicode text as input and return the rendered text in a format compatible with requirements of the Symbian operating system.

3) Deployment and Testing Platforms

Both components of the solution were deployed and tested on the following platforms.

- **WINSCW**
  This is a simulator for theS60 Symbian platform included in Symbian S60 3rd Edition Feature Pack 2 SDK v1.1.2 for Windows Platform.

- **Nokia E51 (A Symbian Phone)**
  The following are the specifications of the Nokia E51 handset—a Symbian based phone:
  i. Symbian: v9.2 S60 v3.1 UI
  ii. CPU: ARM 11 369 MHz Processor
  iii. RAM: 96 MB

G. Testing Results

The SMSLocalized application and language specific modules of Pangocairo framework are deployed and tested on both a Windows emulator (Symbian S60 3rd Edition) and a real device (Nokia E51). The application works successfully on both platforms. Figure 3 shows the SMSLocalized application running on the Nokia S60 3rd Edition Emulator. The on-Screen Urdu Keypad in Nafees Nastalique Open Type Font can also be seen. Figure 4 shows Urdu text written in Nafees Nastalique font (an Open Type Font) as rendered on the Nokia E51.

An Open Type Font file contains glyphs and rules. The glyph tables are in a similar format to those used to store vectorized outlines for TTF files. In addition, rules for glyph positioning and their contextual substitution are represented in different tables. Finally, marks which are associated with glyphs can also be adjusted through rules for finer tuning of fonts. All of these aspects are thoroughly tested for Nafees Nastalique, and the open Urdu font freely available online. More than 500 Urdu ligatures[1] consisting of two to eight characters are chosen from the list of valid ligatures available online (CRULP 2009). The arbitrary selection includes complex ligatures, which exhibit cursiveness, context sensitive shaping and positioning of glyphs. Table 1 shows the ligature counts for two to eight character combinations selected for this testing.

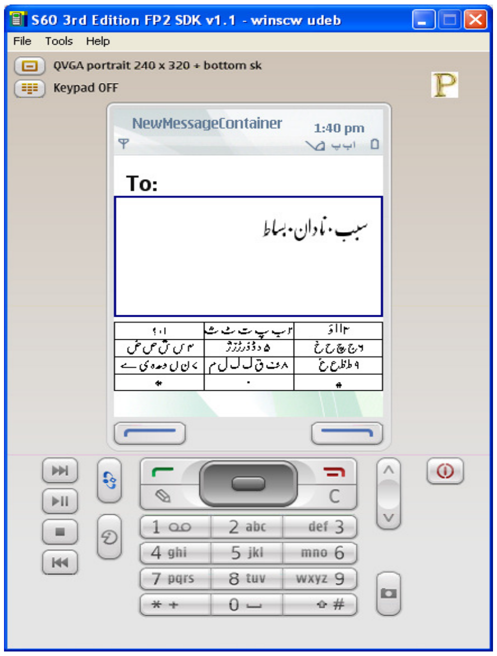
S60 3rd Edition FP2 SDK v1.1 - winscw udeb
File Tools Help
QVGA portrait 240 x 320 + bottom sk
Keypad OFF
NewMessageContainer
1:40 pm
To:

Figure 3: the SMSLocalized Application on Nokia S60 3rd Edition Emulator.

Table 1: Summary of Ligature Set Selected for Testing

Character Count per Ligature	Number of Ligatures Tested
2	90
3	107
4	95
5	81
6	98
7	65
8	20

[1] Ligature is the portion of the written representation of a word that is formed by characters combining together. A word may have one or more ligatures and a ligature may be formed by one or more characters. A non-joining character or a word-ending will end a ligature.

The ligature set included all available Urdu characters. Table 2 shows the frequency of each letter in the test set and the contexts (initial, medial, final and isolated) in which it has been tested. In addition, the mark association and placement is tested. Though the current tests do not test every possible shape of each Urdu letter, as there is glyph variation based on other characters in the context and not just the four contexts listed, the testing is still representative and these results can be extrapolated to un-tested substitution and positioning rules with confidence. The shaded cells in the table are for non-joining characters, which do not occur in initial or medial positions. The ligatures were displayed and manually tested on the Symbian S60 Emulator (WINSCW) and the Nokia E51device.

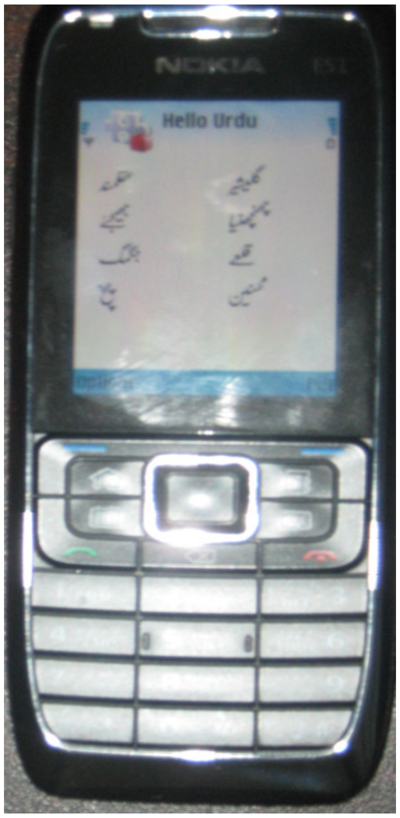

Figure 4: Pango Urdu (Open Type Font Nafees Nastalique) text rendering on a Nokia E51

Table 2: Context and Distribution of Urdu Characters in the Test Set of 500 Ligatures

Character	Frequency	Context			
		Initial	Medial	Final	Isolated
ا	131			97	20
ب	115	65	36	2	12
پ	113	39	63	4	7
ت	177	13	135	18	11
ٹ	102	18	71	4	9
ث	11	2	3	3	3
ج	53	33	17	1	2
چ	69	30	33	3	3
ح	26	11	9	5	1
خ	13	5	4	2	2
د	18			11	7
ڈ	14			10	4
ذ	7			6	1
ر	18			13	5
ڑ	4			2	2
ز	7			3	4
ژ	4			3	1
س	90	24	58	3	5
ش	35	16	8	7	4
ص	14	4	7	1	2
ض	9	3	2	2	2
ط	19	6	9	2	2
ظ	11	3	5	2	1
ع	22	5	12	2	3
غ	12	5	4	2	1
ف	26	9	13	1	3
ق	14	4	6	1	3
ک	98	24	61	2	11
گ	61	23	30	3	5
ل	138	26	101	5	6
م	80	31	35	5	9
ن	254	42	172	19	21
ں	9			6	3
و	21			13	8
ہ	69	7	17	26	19
ھ	176	3	168	5	
ء	5				5
ی	308	6	228	61	13
ے	131			119	12

Figures 4 and 5 show the rendering results of some of the selected ligatures on the phone and emulator respectively, showing the cursiveness, glyph substitution, glyph positioning and mark placement complexities.

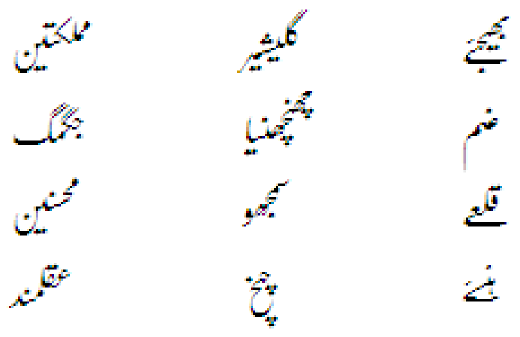

Figure 5: Pango Urdu (Open Type Font Nafees Nastalique) text rendering on Nokia S60 Emulator

After display, all the ligatures were manually inspected for correct shaping, substitution and mark placement. Where there are potential ambiguities, the same are compared with the rendering on the computer to see whether it is the source rendering or the font rules. Detailed testing shows that there are no errors which can be attributed to the porting of these script-specific modules of Pango, verifying completely accurate porting for the module for Arabic script as used for the Urdu language.

The Khmer and Indic modules have also been compiled and tested using limited text. Though no errors have been found, more extensive testing is required for complete verification, so these testing details are not reported at this time. Figure 6 shows Urdu, Devanagari (using the Indic module), and Khmer rendered on Symbian S60 3rd edition emulator.

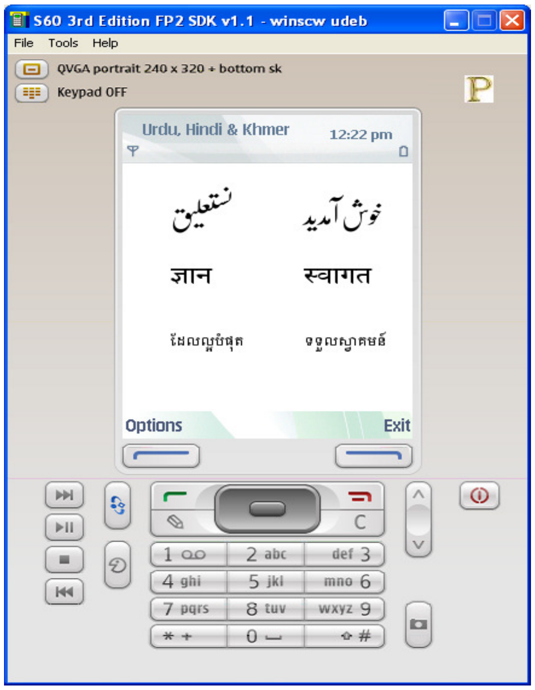

Figure 6: Urdu, Devanagari, and Khmer rendered on Symbian S60 3rd edition emulator.

IV. Conclusion

The global penetration of smart-phones is making local language support for them both urgent and significant, as an increasing number of mobile users want the devices to access local language content. However, we have learnt that smart-phones are still far from current desktops in their support for the local language scripts of developing Asia. The Symbian platform, among the oldest and mature mobile platforms, does not provide complete Open Type Font (OTF) support. However, the porting of Pango script-specific modules can add OTF support to Symbian. This has been successfully achieved through our project. All of the Pango language script modules have been ported to the Symbian OS, with extensive testing carried out for Urdu and initial testing performed for Khmer. Through the process, we have learnt that the Urdu, Indic and Khmer language modules of Pango work well on the Symbian platform. We believe that given the extensive support for international languages by Pango, it is a good choice for serving as a text layout and rendering engine for smart-phone devices.

Currently, the project is continuing to port and test additional script modules. The SMSLocalized application is being integrated to communicate with Pango for rendering and additional work is underway to develop similar support for the Android open source platform.

Acknowledgements

This work has been supported by the PAN Localization project (www.PANL10n.net) grant by IDRC Canada (www.idrc.ca), administered through Center for Language Engineering (www.CLE.org.pk), Al-Khawarizmi Institute of Computer Science, University of Engineering and Technology, Lahore, Pakistan.

References

adMob (2010) AdMob Mobile Metrics [online], available: http://metrics.admob.com/ [accessed 15 Aug 2010]

Android Developers on Google Groups (2010) Localization [online], available: http://groups.google.com/group/android-platform/browse_thread/thread/8887a2fe29c38e7 [accessed 17 Aug 2010].

Apple (2010) iPhone 4 Technical Specifications [online], available: http://www.apple.com/iphone/specs.html [accessed 20 Aug 2010].

Cairo Graphics (2010) Cairo Tutorial [online], available: http://cairographics.org/tutorial/ [accessed 12 May 2009].

Cairo Graphics (2009) Cairo for Symbian OS [online], available: http://code.google.com/p/cairo-for-symbian/ [accessed 18 May 2009].

CRULP (2009) Valid Ligatures for Urdu [online], available: http://www.crulp.org/software/ling_resources/UrduLigatures.htmhttp://www.forum.nokia.com/ [accessed 11 Mar 2010].

Edwards, L. and Barker, R. (2004) Developing S60 Applications: A Guide for Symbian OS C++ Developers, U.S.: Addison Wesley.

Expat (2009) The Expat XML Parser [online], available: http://expat.sourceforge.net/ [accessed 13 May 2009].

Free Type (2009) The FreeType Project [online], available: http://www.freetype.org/index2.html [accessed 12 May 2009].

FontConfig (2009) User's Manual [online], available: http://fontconfig.org/fontconfig-user.html [accessed 13 May 2009].

Forum.Nokia (2009) Support for Open Type Fonts [online], available: http://discussion.forum.nokia.com/forum/showthread.php?163031-Support-for-Open-Type-Fonts [accessed 16 Aug 2010].

Forum.Nokia Users (2009), Discussion Board [online], available: http://discussion.forum.nokia.com/forum/ [accessed 7 Oct 2009].

Google (2009) Localizing Android Apps [DRAFT] [online], available: http://groups.google.com/group/android-developers/web/localizing-android-apps-draft [accessed 14 May 2010].

Google (2010) Android 2.2 Platform [online], available: http://developer.android.com/sdk/android-2.2.html [accessed 10 Oct 2010].

Google Android Community (2010) Arabic Language Support [online], available: http://code.google.com/p/android/issues/detail?id=5597&colspec=id%20type%20status%20owner%20summary%20stars [accessed 19 Aug 2010].

Harrison, R. and Shackman, M. (2007) Symbian OS C++ for Mobile Phones: Application Development for Symbian OS v9, England: John Wiley & Sons, Ltd.

Hussain, S.(2003). 'www.LICT4D.asia/Fonts/Nafees_Nastalique.' Proceedings of 12th AMIC Annual Conference on E-Worlds: Governments, Business and Civil Society, Asian Media Information Center, Singapore.

International Telecommunication Union (2010) ITU sees 5 billion mobile subscriptions globally in 2010 [online], available:

http://www.itu.int/newsroom/press_releases/2010/06.html [accessed 18 Aug 2010].

Kblog (2009) Arabic Language in Android [online] available: http://blog.amr-gawish.com/39/arabic-language-in-android/ [accessed 19 Aug 2010]

Microsoft (2009) OpenType Specification [online], available: http://www.microsoft.com/typography/otspec/ [accessed 10 Oct 2010].

Microsoft (2010) Creating a Complex Scripts-enabled Run-Time Image [online], available: http://msdn.microsoft.com/en-us/library/ee491707.aspx [accessed 16 Aug 2010].

MobiThinking (2010) Global mobile stats: all latest quality research on mobile Web and marketing [online], available: http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats [accessed 16 Aug 2010].

Monotype Imaging (2010) Products and Services [online], available: http://www.monotypeimaging.com/productsservices/ [accessed 5 Aug 2010].

Morris, B. (2007) The Symbian OS Architecture Sourcebook: Design and Evolution of a Mobile Phone OS, England: John Wiley & Sons, Ltd.

Pango (2009) Pango Reference Library [online] available: http://library.gnome.org/devel/pango/stable/ [accessed 15 May 2009].

ParaGon Software Group (2010) Language Extender for Windows Mobile Pocket PC [online], available: http://pocket-pc.penreader.com/ [accessed 16 Aug 2010].

ParaGon Software Group (2010) PILOC for Palm [online] available: http://palm.penreader.com/ [accessed 24 Aug 2010].

Pixman (2009) Pixmann [online], available: http://cgit.freedesktop.org/pixman [accessed 13 May 2009].

Roelof, G. (2009) LibPng for Windows [online], available: http://gnuwin32.sourceforge.net/packages/libpng.htm [accessed 15 May 2009].

Roelof, G. (2009) LibPng [online], available: http://www.libpng.org/pub/png/libpng.html [accessed 15 May 2009].

Sales, J. (2005) Symbian OS Internals: Real-time Kernel Programming, England: John Wiley & Sons, Ltd.

Taylor, O. (2004) 'Pango, an open-source Unicode text layout engine,' 25th Internationalization and Unicode Confernece, Unicode Consortium, Washington DC.

Taylor, O. (2001) Pango: Internationalized Text Handling [online], available: http://fishsoup.net/bib/PangoOls2001.pdf [accessed 10 Jun 2009].

Wali, A., Hussain, S. (2006) 'Context Sensitive Shape-Substitution in Nastaliq Writing system: Analysis and Formulation,' Proceedings of International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE2006).