The Research and Implementation of Mongolian Extension to Swing and Android UI Components

ZHANG Zhongwei, LIN Min

College of Computer and Information Engineering , Inner Mongolia Normal University, Hohhot, China
joyzzw@yeah.net, cieclm@imnu.edu.cn

Abstract - Since the traditional display layout of Mongolian has its particularity and the relatively lagging of Mongolian character set included in the international coding standard, current mainstream development platforms have neither a completed set of UI components support Mongolian features, nor any well-developed Mongolian UI components extension method. This paper find out a reasonable UI components extension method support Mongolian features by study the architecture of parsing Java Swing and Android UI components. In addition, we use this method extend the commonly used Swing and Android UI components. Results show that the extended components have the display and edit functions support the international coding standard and conform to Mongolian features, consequently, it can meet the need of Mongolian application development over the Internet and mobile platforms.

Index Terms - Swing Components, Android UI, Component Extension, Mongoliant

1. Introduction

Mongolian information processing technology started earlier, and have made certain progress in both theory and application level. However, due to the Mongolian characters haven't enter the era of international standard codes until 2000, is lagging behind; the layout of traditional Mongolian has its particularity that it requires input from top to bottom and from left to right; and the traditional Mongolian used mainly in Inner Mongolia with geographic limited[1]. These peculiarities result in the UI components of current mainstream development platform support poorly for Mongolian, which seriously affect the development and application of Mongolian software. In addition, although smart phones and mobile devices become more and more popular, the UI components of current mobile platform support very inadequately for Mongolian, Mongolian application software is very rare, which bring a very negative impact on Mongolian users in the internet era and adversely affect Mongolian culture heritage.

Swing is one of the mainstream UI components in Java technology, with the benefits of mature technology, complete function, and widely used[2]. Develop a set of Swing-based Mongolian Java GUI components will help to reduce Mongolian application development costs and improve the efficiency of software development. Android is a mainstream mobile development platform launched by Google company. Android UI system lies in application framework layer, which use Java development language entirely[3-5]. Therefore, the extension method to Java Swing components also applies to Android UI components.

2. Related Technologies Overview

A. Swing architecture

Swing is based on something called a "modified MVC (model-view-controller) architecture". Based on this architecture, each swing component contains a model and an UI delegate. UI delegate is responsible for painting screen and handling GUI events. Model is in charge of maintaining information or states of the component. Standard Swing components include at least three basic objects [2]: Component, Model and UI Delegate.

The component object is the center of Swing components, Model and UI Delegate are integral part of it. It is responsible for providing some of the APIs and thus coordinates and controls Model, UI Delegate and Renderer object.

The model object is responsible for storing the state of the component and is divided into two categories: GUI-state models and application-data models[6]. GUI state models are interfaces that define the visual status of a component; An application-data model is an interface that represents some quantifiable data that has meaning primarily in the context of the application. The component object through its set/get methods to access or modify the Model object. The Model object also provides some methods to get/set data and respond to events.

UI Delegate is a class object which has both the view and controller roles. UI Delegate responsible for implementing the component's appearance rendering and event response which is provided by the current LAF packages can be dynamically modified. A look-and-feel implementation provides concrete subclasses for each abstract plaf UI class. The UI delegate is created in the component's constructor and is accessible as a JavaBeans bound property on the component. A view object will be created according to the component's type and the information of element object when the UI delegate is created. This view object is responsible for displaying and editing the text of component.

B. Principle of Swing component's text rendering and text editing

Swing using the MVC architecture separates data and view. For those text related component, their text data is hold in the corresponding Document object, which is to say, Document is the concrete implementation of the model role played in Swing MVC architecture. To display the text of the component, UI Delegate of that component firstly get the text data from Document object, then draw the text data onto the screen using the specific styles provided by UI Delegate. Java programs

must use this Graphics object or one derived from it to render output. When the component text has been modified, what is really changed underground is Document content[7].

The text component's editing function is achieved through the event listeners. There are two main event listener interfaces which are associated with the text editing function[8-9]: one is Undoable Edit Listener interface, and other is Document Listener interface. The Undoable Edit Listener interface monitor UndoableEvent event and records all operations of the text component in order to assist in the implementation of undo and redo commands. The Document Listener interface monitor DocumentEvent event (such as type characters, delete characters, cut, paste, etc.), and triggers the text component's repaint method to update the display of text. With these two interfaces, text components can be achieved text editing functions.

C. The Mongolian international standard encode and methods to map nominal characters to displayed characters

The Mongolian international standard encoding has been accepted by Unicode technology committee. In the standard of ISO/IEC 10646, 176 code positions are provided for Mongolian, in which only 35 are occupied by traditional Mongolian characters [10-12]. In this standard, only some abstract Mongolian characters are stored according to Mongolian pronunciation, and these are called nominal characters. Different from ordinary pinyin words, one Mongolian character's shape, which is called displayed characters, will change with its position in a word and the word's part of speech and so on. Because no Mongolian displayed characters are stored in ISO/IEC 10646 standard, when displaying Mongolian, the nominal characters must firstly be mapped to their correct displayed characters according to their contexts.

Nowadays, there are two methods to map nominal characters to displayed characters [7]: one is to store displayed characters and their customized codes, which normally occupy the private area PUA, in the TrueType font, then, according to the features of Mongolian, write a mapping program to analyze Mongolian characters' contexts in order to get their correct displayed characters' customized codes, and finally access the glyph data in the TrueType font. The advantage of this method is not depending on operation system, and the disadvantage of it has to design a special algorithm on the layer of application program to implement the rule of mapping nominal characters to displayed characters, which is lowly general. Another way is by using the OpenType font technology to store displayed characters and the rule of mapping in the font, and to analyze these rules by special modules of the operation system in order to get the correct displayed characters' ID (glyphID). The advantage of this method is not requiring the support of a mapping program in application layer and is highly general, but the disadvantage is that some system do not support the analyzing of OpenType font untill now, which makes this method hard to be applied across platforms. This paper used the first method.

3. The Mongolian Extension Method of the Swing Components

A. The reason why Swing components do not support Mongolian text display

Ordinary Swing components can not support the Mongolian text which is reflected in three aspects:

First, Swing components can not display Chinese and Mongolian text at the same time that when the text contain both Chinese and Mongolian characters, one of the characters will not be displayed. Figure 1 is a JTextArea Component test case, the first JTextArea component uses the default font and the output show that Chinese characters are displayed correctly in that, whereas the Mongolian ones don't; Second JTextArea component uses the Mongolian White font and the output show that Mongolian characters are displayed correctly in that, whereas the Chinese ones don't. The fundamental reason for this result is that: Swing components use this Graphics object or one derived from it to render output. Graphics object at a time can be set only one font. When the font of Graphics object is set to Chinese font(the default font), Swing components can not display Mongolian characters correctly. The reason is that Mongolian characters can not find its Corresponding glyph in the Chinese font. Similarly, when the font of Graphics object is set to Mongolian font, Chinese characters will not be displayed correctly.

Second, even if the font of Graphics object is set to Mongolian font, Swing components' display of the Mongolian characters is also wrong. Because the Mongolian characters to be displayed are no be converted from nominal characters to displayed characters, so the display is just the glyph of the Mongolian nominal characters. For example, the second test output in Figure 1. Swing components' runtime platform is JVM which does not support OpenType font until now. Meanwhile there is not a mapping program to convert Mongolian nominal characters to displayed characters. So the Mongolian characters can not be displayed correctly.

Third, Swing components do not support the Mongolian characters to be displayed and edited vertically. Since the layout of Mongolian text has its particularity that it requires input and display from top to bottom and from left to right. However there is no specific interface to support the Swing components to input and display text vertically. So this is an aspect that Swing components can not support Mongolian text.

Fig 1. JTextArea component test case

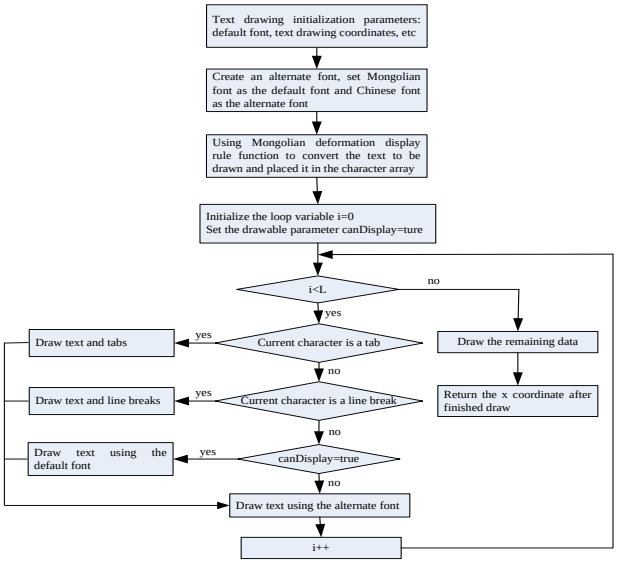
B. Extend the Swing components

Against the three aspects of the Swing components which are responsible for the incorrectly displaying of Mongolian, we

also extend the Swing components from three aspects. These three aspects seem totally independent, however, we find that they are connected and related internally when implementing of the technology details. For example, When we extend the Swing components' capabilities to make it display and edit both Chinese and Mongolian text at the same time, we both must consider both the problem of Graphics object' font and the problem of Mongolian deformation display.

1) Extend the Swing components to display both the Chinese and Mongolian text at the same time

By exploring and analyzing of the source code, we find that Swing components needs to engage multiple level methods invocation chain to render the text correctly. However, many work flows share a common procedure at the last part of that invocation chain, so we could display the Chinese and Mongolian text at the same time, side by side, through the extending to the commonly shared procedure. The rendering process after extending is shown in Figure 2.



Note: L is the length of character array

Fig 2. the rendering process after extending

Extension is mainly manifested in two aspects: First, before rendering the characters, convert Mongolian nominal characters to displayed characters with the mapping program(this program will be explained later); Second, set an alternative font and make sure the default font is Mongolian font, alternate font is a Chinese font. When the component renders its text, it must use the canDisplay() method of the default font object to determine whether the current character can be displayed, at first. If the current character can be displayed then render this character with the default font object, otherwise render this character with the alternative font. Through this extension the function of displaying Chinese and Mongolian text at the same time can be achieved.

2) Mapping Mongolian nominal characters to displayed characters

So far, JVM can not parse the rules in layout table which is part of OpenType font. In order to implement the application across platform, this paper chooses the first method mentioned above to map nominal characters to displaying characters. Concrete steps are: First of all to establish a special OpenType font. This font, besides the rules of mapping Mongolian nominal characters to displayed characters, also has all of the Mongolian displayed characters' glyph. These glyph data are encoded to Unicode's private area (PUA). Then this font can be used at any system platform, on matter whether the platform supported the OpenType font or not. Secondly, write a mapping program with Java language to analyze Mongolian character's context in order to get the correct displayed character. When the text needs to be displayed, we use this mapping program to get correct displayed characters, and finally access the glyph data in the special OpenType font. Then the Mongolian text will be displayed in correctly.

3) Extend the Swing components to display and edit the Mongolian characters vertically

Swing components edit and display all content via EditorKit and EditorKit displays text using view class. So for, we focus on extending the view class to implement displaying text vertically. In the view class, most of the methods will use a lot of coordinate and size parameters, when they display text. The horizontal and vertical positions in displaying from top to bottom and left to right are symmetrical about the line y=x. So when we extend the method of the view class, we just need to swap and appropriate adjustment some parameters which are associated with displaying and editing text.

There are four main steps to achieve the goal of displaying and editing text vertically: Step 1: extend the setSize() method of the view class, as well as the methods related to component layout. In this step we will swap and store the width and height of the component; Step 2: extend the paint() method and the associated methods. In this step we should use the Graphics2D object's rotate() method to rotate the text vertically and modify some positional parameters associated with the text displaying; Step 3: extend the getNextVisualPositionFrom() method to change navigation by keys strategy. since the vertical text is displayed from top to bottom and from left to right, so left and right("←" and "→") arrows should go to previous and next row, up and down("↑" and "↓") arrows accordingly will move caret to previous or next char; Step 4: extend the modelToView() and ViewToModel() method to display the caret horizontally. Real caret rectangle for each caret position is returned by modelToView() method. We extend the method and change shape returned by super. Then we modify viewToModel() method changing coordinates to allow proper caret setting in a point which was clicked by user. After the extension by the above four steps, the text can be displayed and edited vertically.

We take the PlainView class as the example, and to describe the specific content to be extended of each method. PlainView's extension step is shown in Table 1.

Table 1 Mongolian vertically display and edit extension step of javax.swing.text.PlainView class

Step	Extension Method	Extension Content
1	setSize(float, float)	Assign width, height to height and width, the two properties of View class, respectively, implement the exchange.
	getMaximumSpan(int)	Set judgment conditions, case View.X_AXIS: getMinimumSpan(View.Y_AXIS); case View.Y_AXIS: getMinimumSpan(View.X_AXIS);
	getMinimumSpan(int)	
2	paint(Graphics, Shape)	Rotate *Graphics* by 90 degrees to make the characters display rotate 90 degrees; Exchange the x, y coordinates of text outsourcing rectangular frame *alloc* as well as width and height; Exchange blank above *linesAbove* and blank below *linesBelow*; Text drawing area *lineArea = lineToRect(a, linesAbove)*, the second parameter changed to *linesBelow*; Initialize *y = lineArea.y - fontHeight + metrics.getAscent()+2*; Initialize *line* with *linesBelow* in statement for (int *line = linesAbove*;); Change y += *fontHeight* in the loop to y -= *fontHeight*;
	damageLineRange(int, int, Shape, Component)	Implement the exchange of x, y coordinates and the exchange of width and height both in Rectangle area0 and Rectangle area1;
3	getNextVisualPositionFrom(int, Bias, Shape, int, Bias[])	NORTH changed to WEST; SOUTH changed to EAST; EAST changed to SOUTH; WEST changed to NORTH; Swap the execute statements of case EAST: and case WEST:;
4	modelToView(int, Shape, Bias)	*alloc = newRectangle((Rectangle)a)*; Exchange the *x, y* coordinates of *alloc*, exchange *width* and *height*;
	modelToView(int, Bias, int, Bias, Shape)	To rectangle *r0*:*r0.x = r0.y*; *r0.y = -tmp - r0.width*/2; Exchange *width* and *height* of *r0*. Make the same modifications to rectangle *r1*;
	ViewToModel(float, float, Shape, Bias[])	Exchange the *x*, *y* coordinates of rectangle *alloc*, exchange *width* and *height*; Exchange int *x* and int *y*;

4. Results

Using the extension method as the paper find out, we have efficiently achieved the extension of JLabel, JTextFiled, JTextArea components of Swing. The extended components can not only achieve the goal to correctly display Chinese and Mongolian text at the same time, but also can display and edit the text vertically. The result is shown in the left of figure 3.

Fig 3. the extended Swing and Android UI components

Using the same extension method, we have also achieved the extension of TextView and EditText components of Android platform. The extended components can not only achieve the goal to correctly display Chinese and Mongolian text at the same time, but also can diplay and edit the text vertically. The result is shown in the right of figure 3.

5. Conclusion

The paper researches mainly on the Mongolian extension method of the Swing and Android UI components, thus finding out an reasonable extension method, and actually achieving the goal to extend JLabel, JTextFiled, JTextArea components of Swing and TextView, EditText components of Android based on the method, the test results show that the extension method is stable and efficient. The extended components can achieve the goal to correctly display Chinese and Mongolian text at the same time but also can display and edit the text vertically, the implementation of these components can cut down the development cost of the PC and Android Mongolian software interface based on JAVA, thus improving the development efficiency of the software, and using the components at the same time can also make the former software client based on Swing and software interface based on Android completely show Mongolian style, thus it can meet the need of Mongolian application development over the Internet and mobile platforms.

References

[1] Amy Fowler. Painting in AWT and Swing. (2003)[2013-05]. Online-Dokument: http://www.oracle.com/technetwork/java/painting-140037.html.

[2] Loy M, Eckstein R, Wood D, et al. Java Swing.[S.l.]: O'Reilly, 2012: 13-16.

[3] Gong Lei, Zhou Cong. Design and Research of Mobile Terminal Application Program Based on Android. Computer and Modernization: 2008, (8): 85-89.

[4] Ding Ershuai. Android environment AppWidget Architecture. Xi An University of Electronic Science and Technology, 2011.

[5] Luo Shuyuan. Android Widget System Design and Implementation. Beijing Jiaotong University, 2012.

[6] Dai Xin. Developement of Java Swing Program. Software Guide: 2007, (9): 138-139.

[7] Yao Yandong, Wu Jian, Sun Yufang, etc.. Research and Realization of Tranditional Mongolian Deformation and Displaying Mechanism. Journal of Chinese Information Processing: 2005, 18(5): 84-89.

[8] ITEEDU. Event Dealing of JTextArea (2012)[2013-05]. http://www.iteedu.com/plang/java/jtswingchxshj/49.php.

[9] Hu Jiafen. Study on the Multi-Thread Mechanism of Swing Graphic User Interface. Computer Konwledge and Technology, 2012 (31): 7481-7482.

[10] International Standard ISO/IEC 10646-1 Second Edition. Information technology-Universal Multiple Octet Coded Character Set(UCS), 2000.

[11] Que Jingzhabu. Mongolian Encoding. Hohhot: Inner Mongolia University Press, 2000.

[12] Gong Zheng. Research of Mongolian Encoding Transforming. Hohhot: Inner Mongolia, 2007.