A Proposal for a Japanese Keyboard on Cellular Phones

Maurice Margenstern[1], Benoît Martin[2], Hiroshi Umeo[3], Shogo Yamano[3], and Kazuhiro Nishioka[3]

[1] Université Paul Verlaine − Metz, IUT de Metz,
LITA, EA 3097, UFR MIM, Île du Saulcy,
57045 Metz, Cédex, France
margens@univ-metz.fr
[2] Université Paul Verlaine − Metz,
LITA, EA 3097, UFR MIM, Île du Saulcy,
57045 Metz, Cédex, France
benoit.martin@univ-metz.fr
[3] Faculty of Information Science and Technology,
Osaka Electro-Communication University,
Neyagawa-shi, Hatsu-cho, 18-8, Osaka, Japan
umeo@cyt.osakac.ac.jp

Abstract. In this paper, we propose a new way to display alphabetical signs in order to write messages on a cellular phone in the Japanese language.

1 Introduction

In [1], the second author tackled the problem of defining a good way to put messages on a cellular phone in a very new approach. The idea is to use the pentagrid, a tiling of the hyperbolic plane, thoroughly studied from an algortihmic point of view by the first author, see [5]. The pentagrid was already used by the first and the second author in order to define a colour chooser, see [2].

The approach of [1] was confirmed by tests, however performed on a small number of persons.

In the next section, we remind the minimal features of hyperbolic geometry in order the reader could understand the basic material of the paper. In the following section, we describe the proposal. Then, we describe the first results on the work for implementing the proposal on cell phones.

2 A Few Words on the Pentagrid

We use the model of the hyperbolic plane devised by Poincaré in the last third of the 19$^{\text{th}}$ century. In this model, the hyperbolic plane is the set of points which lie in the open unit disc of the Euclidean plane whose border is the unit circle. The lines of the hyperbolic plane in Poincaré's disc model are either the trace

H. Umeo et al. (Eds): ACRI 2008, LNCS 5191, pp. 299–306, 2008.

of diametral lines or the trace of circles which are orthogonal to the unit circle, see figure 1. We say that the considered lines or circles **support** the hyperbolic line, h**-line** for short and, most often, simply **line** when there is no ambiguity.

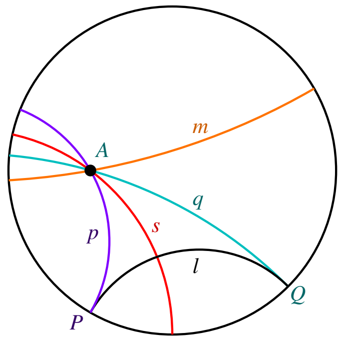

Fig. 1. The lines p and q pass through A and are parallel to the line ℓ, with points at infinity P and Q, on the border of the unit disc. The h-line m passes through A too and is non-secant with ℓ: it does not cut it, neither in the unit disc nor outside.

The angle between two h-lines are defined as the Euclidean angle between the tangents to their support. This is one reason for choosing this model: hyperbolic angles between h-lines are, in a natural way, the Euclidean angle between the corresponding supports. In particular, orthogonal circles support perpendicular h-lines.

A striking property of this geometry is that there is no rectangle. From this, one can prove that two lines of the hyperbolic plane are non-secant if and only if they have a common perpendicular.

Contrary to the Euclidean plane were there are only three kinds of tilings based on the recursive replication of a regular polygon by reflection in its sides and of the images in their sides, here there are infinitely many such tilings. This was proved by Poincaré in late 19$^{\text{th}}$ century. In one of these tilings, called the **pentagrid**, the regular polygon is a pentagon with right angles.

Figure 2 sketchily remembers that the tiling is spanned by a generating tree. Now, as indicated in figure 3, five quarters around a central tile allows us to exactly cover the hyperbolic plane with the **pentagrid**.

In the right-hand side picture of figure 3, we remember the basic process which defines the coordinates in a quarter of the pentagrid, see [5]. We number the nodes of the tree, starting from the root and going on, level by level and, on each level, from the left to the right. Then, we represent each number in the basis defined by the Fibonacci sequence with $f_1 = 1$, $f_2 = 2$, taking the maximal representation, see[3,5].

The main reason of this system of coordinates is that from any cell, we can find out the coordinates of its neighbours in linear time with respect to the coordinate of the cell. Also in linear time from the coordinate of the cell, we can

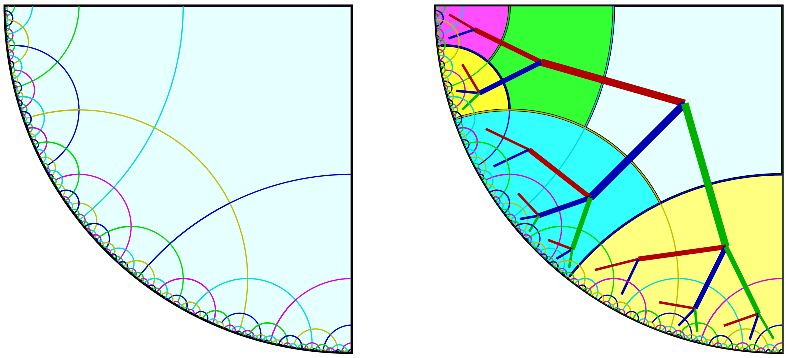

Fig. 2. On the left: the tiling; on the right: the underlying tree which spans the tiling

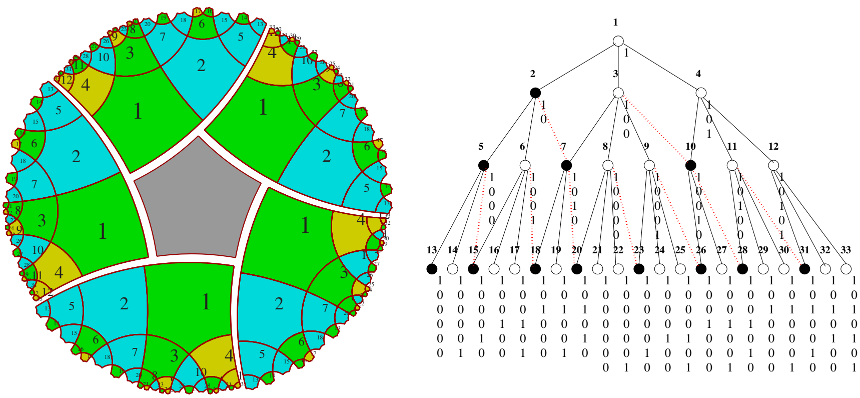

Fig. 3. On the left: five quarters around a central tile; on the right: the representations of the numbers attached to the nodes of the Fibonacci tree

compute the path which goes from the central cell to the cell. Further references on this system and on hyperbolic geometry can be found in [5] and [4].

Now, as the system coordinate is fixed, we can turn to the application to keyboards for cell phones.

3 The Proposal

The proposal is based on the idea to use the pentagrid to display **hiraganas** and **katakanas**, the syllabic alphabets of the Japanese language.

Looking at the pentagrid of figure 4, does not it seem that the pentagrid is fit for the Japanese language? Indeed, the phonetic writing of Japanese, using **hiraganas** and **katakanas** is based on ten series, in which the letters are ordered

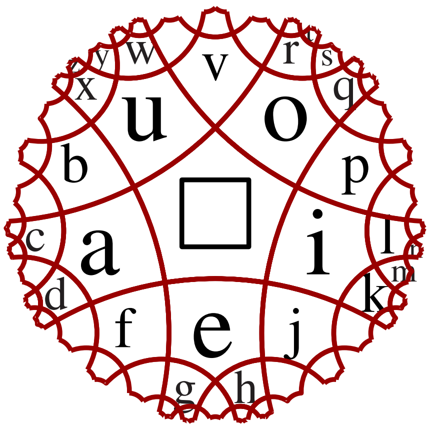

Fig. 4. The proposal of keyboard from [1]

in relation to the vowels they contain. Now, the Japanese language has five basic vowels, **a**, **i**, **u**, **e** and **o**, following the traditional Japanese order, the corresponding hiraganas being: あ , い , う , え and お .

This keyboard is displayed in figure 5, where each sector, spanned by a Fibonacci tree, is devoted to the syllables associated to the same vowel. The corresponding vowel is put at the root of the tree. The others are displayed level by level, following the traditional order while going from the left-hand side border to the right-hand side one of a sector.

However, the project of this keyboard, which we are developping does not stop on this representation of the hiraganas. Of course, a similar keyboard may display the katakanas, as their traditional order is exactly the same as that of the hiraganas. Now, standard Japanese language makes use of Chinese characters named **kanjis**. The main goal of the project is to give access to the kanjis themselves, using the display of the Japanese keyboard, see figure 5.

Indeed, each kanji has several pronunciations represented by a hiragana or a katakana. The idea is to go from the pronunciation to the kanjis. Now, several kanjis have the same pronunciation and the same syllabic sign represents several kanjis. Nevertheless, this system seems to be convenient: once a hiragana/katakana is selected, a new viewer displays the possible kanjis inside a pentagrid view. In such a display, the kanjis can be ordered by the number of strokes as in traditional dictionaries.

4 For the Cell Phones

The project is still under progress. Below, we can see pictures taken from an experiment performed in Japan under the supervision of the third author. In figure 6, we can see a snap shot of the screen. On the screen, the different parts

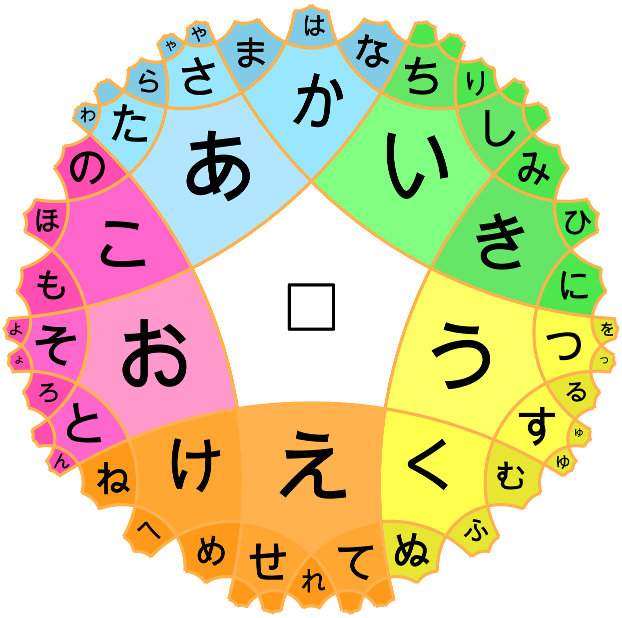

Fig. 5. The Japanese keyboard

of a mobile phone are represented, in particular the screen of the mobile phone. In the display of the figure, the hiraganas are displayed in the reverse order with respect to the order of figure 5.

But the work did not stop with tests on a computer. It is now on the implementation on an actual mobile phone. Figure 9 displays a picture of what can be seen by a user of an actual mobile phone on which the keyboard with the pentagrid appears on its screen.

Our VirHKey is implemented on a platform of *iappli Development Kit for DoJa (FOMA) Version 3.00* which is a Java-based software development workbench supplied with NTT Docomo Inc. for cell phone. Figure 6 is a screen snapshot of the application software.

Figure 7 illustrates some functions for inputting Japanese characters using the VirHKey implemented on the cell phone. It also gives a correspondence between a screen and keys on the cell hpone. Each key indicated by numbers operates as follows:

- **1**: **terminate** the VirHKey
- **2**: **input** a character displayed at center cell
- **3**: **change** a default input mode into a special character input mode
- **4, 5, 6, 8, and 9**: **move** a character around into the center cell
- **7**: **display** characters being input
- **10**: **delete** one character

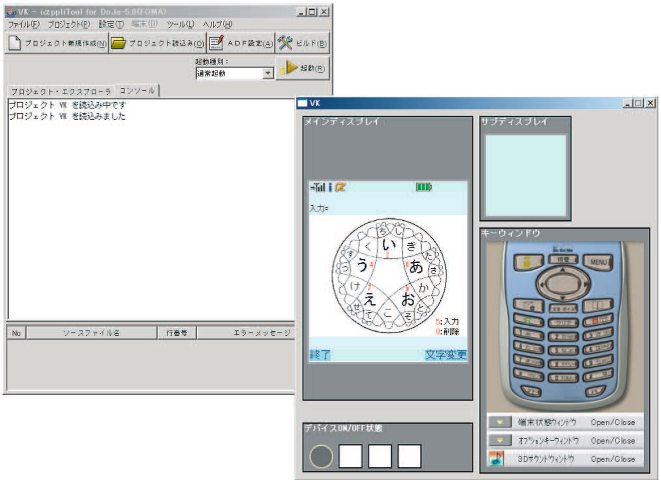

Fig. 6. Screen snapshot of *iappli Development Kit for DoJa* developing the VirHKey

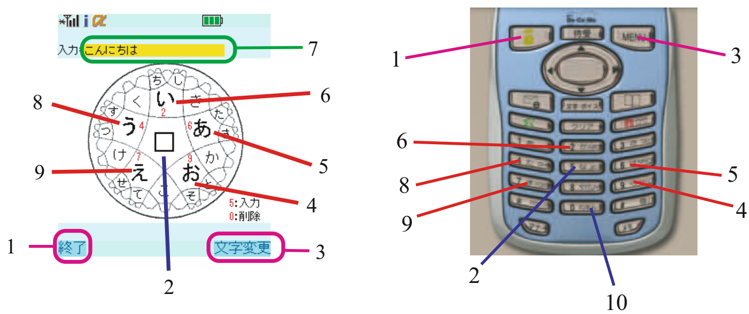

Fig. 7. Screen snapshot of *Functions for inputting Japanese characters using the VirHKey*

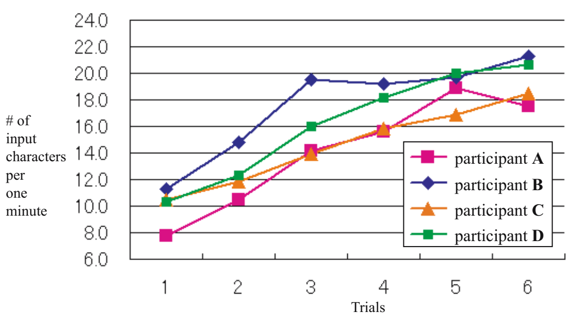

Fig. 8. Screen snapshot of *Usability test for inputting Japanese characters using the VirHKey*

Fig. 9. The Japanese keyboard on an actual cell phone, from an experiment at the University of Electro-communications of Osaka

We have made a small usability experiment for inputting Japanese character using the VirHKey. See figure 8. The vertical axis shows the number of characters input for one minute by each testee. It is noted that input errors are excluded. The horizontal axis means the trials. As can be seen, input performance at each testee is growing progressively to about twice in comparison with the initial performance, however, its absolute performance is not so good in comparison with the standard Japanese character input scheme. The biggest reason is that all of the testees are not familiar with the character arrangement in VirHKey. The Japanese consists of many characters including alphabets in a usual communications on mobile phones. We usually have to hit several keys to get a desired character. For example, to select "(ho)" one has to press a key at six times in a standard input scheme. But on VirHkey one has to press a key 9, 2, 4 and key 5 at each stroke, totally four times. The number of strokes is smaller than the standard way, however, we have to change the key at each stroke. The input performance should be improved after trials. We are convinced the VirHkey would be efficient useful for general-purpose intelligent terminals with small display areas and devices where relatively large number of characters must be tackled.

This is very interesting! However, the work does not stop here. As previously mentioned, the project deals with kanjis too. For this, the big problem is the input of all the kanjis and their associations to the hiraganas and katakanas. There are data bases for such associations but the related problems are, of course, beyond the scope of this paper.

5 Conclusion

It seems to us that this project is very promising. The tests on users seems to indicate that the proposed tool is very efficient. We are convinced that the project, completed with the kanji version will meet a great success.

References

1. Martin, B.: VirHKey: a VIRtual Hyperbolic KEYboard with gesture interaction and feedback for mobile devices. In: Proceedings of MobileHCI 2005, Salzburg, Austria, September 2005, pp. 99–106 (2005)
2. Chelghoum, K., Margenstern, M., Martin, B., Pecci, I.: Palette hyperbolique : un outil pour interagir avec des ensembles de données. In: IHM 2004, Hyperbolic chooser: a tool to interact with data, Namur, Belgium (September 2004)
3. Margenstern, M.: New Tools for Cellular Automata of the Hyperbolic Plane. Journal of Universal Computer Science 6(12), 1226–1252 (2000)
4. Margenstern, M.: Cellular Automata and Combinatoric Tilings in Hyperbolic Spaces, a survey. In: Calude, C.S., Dinneen, M.J., Vajnovszki, V. (eds.) DMTCS 2003. LNCS, vol. 2731, pp. 48–72. Springer, Heidelberg (2003)
5. Margenstern, M.: Cellular Automata in Hyperbolic Spaces, vol. 1, Theory, 422 p. Old City Publishing, Philadelphia (2007)