# A TEXT-ENTRY SYSTEM FOR INPUT-CONSTRAINED DEVICES

TOM GIBARA

## 1. Background

I own a 3rd generation iPod. I enjoy using the device but my greatest frustration with it is the inability to search for an artist, song or album in a manner analogous to the search offered by iTunes. For this reason, I became interested in whether a practical text input system exists for what I term 'input constrained devices'. These are devices that may feature relatively powerful processors and good screens, but which are primarily limited by the physical interface that they provide to the user; I do not include devices such as mobile phones which feature a relatively large number of buttons. This document, and its associated software, are the product of a couple of days of investigation into this question.

## 2. Constraints and Considerations

I began by compiling a set of physical UI constraints that the text entry system would need to accomodate:

- Devices may feature a small number of buttons. At a minimum, there could be as few as four: play/pause, forwards, backwards and a generic Action key.
- It is possible that the buttons are arranged as a D-pad[1] so chorded operation is precluded.
- One button should be reserved for exiting the text entry mode.
- Though screens may be capable of displaying color, it is possible that only black & white can be displayed. Even on devices capable of displaying colour, nuanced color reproduction may be impossible.
- Devices are generally mass produced for international markets. Consequently, they may be operated in a wide variety of different locales.

From these constraints we can deduce that the text-entry system must have the following characteristics:

- It is operable with at most three buttons.
- It does not assume that any two buttons can be pressed at the same time.
- It is operable via a black & white screen.
- It must be possible to enter text in a variety of different languages.

In addition to these forced characteristics, I have a personal preference for text-entry systems that do not completely relegate punctuation to a league of second-division characters that are totally impractical to type. I also dislike systems that distinguish between long and short button presses. I find such systems annoying

*Date*: 10th July 2007.

[1]A direction pad common in games console controllers.

when I am a novice typist, since I often dwell on a keypress too long; and more importantly as an advanced typist, since I am required to wait an abitrary length of time until some keypresses are recognized.

Guided by these considerations, I first explored dictionary based approaches (à la T9[2]) based on the principle of one-click one-letter followed by a disambiguation phase (and possibly by a direct entry system for unrecognized letters). Though I think that such a system is workable (I came up with a number of designs) my concern with this approach was threefold:

- Elevating the accuracy of letter selection (to the degree necessary for a system with so few buttons) seemed to require computationally intensive algorithms[3].
- Any such system needs to provide a direct text-entry system in the case where a non-dictionary word is being entered and thus is at least as complex to use/implement as the best direct system.
- Dictionary approaches have limitations if only word segments are being typed-in (as opposed to whole words). This may be the case if a user is performing a general text search (a case that I am directly considering).

So I then turned my attention to designing a direct text-entry system; one in which the user focuses on entering each letter in turn (in contrast to typing whole words).

## 3. The System

The text-entry system I designed operates as follows:

- The system is modal - one button is designated the 'mode' button and is used to change modes.
- Each of the remaining buttons (of which there must be at least two) is assigned a fixed and unique colour.
- Each mode has a keyboard associated with it. Different languages may feature different modes and different numbers of modes.
- The keyboard for the current mode is displayed on screen for as long as the text-entry system is active.
- Each key on the displayed keyboard is assigned a sequence of colours. Depending on the capabilities of the device, the screen may be of sufficiently high quality to display the entire sequence of colours for each key. Alternatively, only one colour might be displayed for each character - the next colour in its sequence.
- Each sequence of colours unambiguously identifies one letter on the keyboard in the form of a self delimiting code. There is no constraint on the length or composition of the sequences; sequences may contain only a single colour; colours may be repeated; sequences may be arbitrarily long.
- As the user presses the colour buttons, a sequence of colours is established which will ultimately match exactly one key on the keyboard. Keyboard displays that render only one colour per key must update the display on each button-press to ensure that the next colour in the sequence is displayed at all times.

[2]T9 is a predictive text input system developed for mobile phones.

[3]Certainly, on the surface, the algorithms appear costly in terms of time and/or memory. However, it might be the case that significant optimizations might be possible.

- Any key that has been eliminated as a possibility (by virtue of its colour-sequence being incompatible with the sequence entered by the user) is displayed as *unavailable*.
- After a key has been completely established, each key on the keyboard is reassigned a new sequence of colours. *This is not necessarily the same sequence as before.*
- Sequences are generated by examining the characters immediately preceeding the caret, estimating the probability of each key on the keyboard given that it is preceeded by those characters, and then assigning each key a colour-code whose length reflects its estimated probability; probable keys are assigned short sequences, improbable keys are assigned long sequences.
- If the user presses the mode key mid-sequence, then the current sequence is aborted and the keyboard is returned to the same state it was in before any colours were specified by the user.
- If the user presses the mode key between sequences, the current mode cycles through all available modes. This has the effect of displaying a new keyboard for each mode. Keys may be repeated in different modes, though their sequences will most probably be different in each mode.

## 4. Pros & Cons

The system has a number of benefits:

- It can be adjusted for various numbers of buttons; at least 3 are necessary (1 mode + 2 colours) but more can be used, though the practical limit is probably 5 (1 mode + 4 colours)
- The system, though modal, has a very simple cyclic modal model that should not confuse users - returning to a known state is always as simple as "keep hitting the mode button" until a recognized keyboard appears.
- There is an enormous amount of flexibility in how the keyboards can be organised. There is no restrictions on the number of keys per keyboard, nor on the number of keyboards that a key appears on, nor on the positions of any key on any keyboard. This makes it very amenable to adaptation for different languages.
- The system is completely insensitive to the speed of operation (assuming that the device's processor and screen can respond sufficiently quickly to the user's actions).
- The system is well suited to operation on a D-pad and other such input devices because it does not require more than one button to be pressed at any given time.
- The system can be adapted to black & white displays by limiting it to those two colours.

Due to the constraints inherent in typing human readable text with so few buttons, the system also has a number of weaknesses:

- Users must look at the keyboard for each character they wish to type in order to know its colour sequence. This places a natural constraint on the typing speeds that could be achieved.
- In order to type efficiently, users must memorize both the colours associated with each button and the position of each key on each keyboard.

- True color systems (as opposed to greyscale color assignments) might pose problems for users who suffer from colour blindness.

## 5. Implementation

To prove the system as being workable, I created a flexible implementation in Java. In this implementation:

- A corpus is used[4] to identify frequent n-graphs (tuples of letters) which are then used to generate probability estimates for a given character at a given point in the text.
- The probabilities are converted into colour sequences using an n-ary huffman encoding.
- Four modes are used: lower-case, upper-case, numbers, and punctuation. The alphabetical keys are arranged in QWERTY order.
- The keyboards are displayed and operated via a component written in the Swing framework.

## 6. Evaluation

I have not attempted to conduct a study into the usability of this typing system, but anecdotal evidence indicates that the system is very easy to learn. For users who are familiar with the QWERTY keyboard, the most significant task seems to be learning the association between colours and buttons. Feedback indcates that users believe they would be more comfortable with fixed color sequences, but this has not been verified.

Ultimately, one needs to consider the input system against a 'natural' input mechanism for a given device. In the context of an iPod, the obvious interface is that of letter selection using the click-wheel to move left and right through the alphabet. For a device that features a D-pad, letter selections could be made by moving a cursor over a displayed keyboard.

On these grounds, the system's true benefits are slightly uncertain. Though it is almost certainly faster than either of these alternatives, I do not believe it can be regarded as equally intutitive.

## 7. Applications

I believe that the system could provide a usable typing mechanism for input-constrained devices. Naturally, no user would want to enter large volumes of text in this way, but for the input of small amounts of text (say into simple digital forms or application search fields) the system seems perfectly adequate.

The system might also prove useful for individuals with accessibility problems that prevent them from using a standard keyboard. The speed-insensitive nature of the system, combined with its small number of required buttons might make it ideal for users who can only rely on gross motor skills for computer-peripheral operation.

[4]Currently a collection of P.G. Wodehouse stories from Project Gutenberg `http://www.gutenberg.org`.

## 8. Improvements

During the implementation of the system, I identified the following possible improvements:

- For the purpose of simplifying the keyboard display, and reducing the demands on a user's short-term memory (or eyesight), it may be sensible to use a length-limited code instead of the unbounded Huffman encoding used in the implementation. Naturally this would be at the expense of decreasing the code's efficiency.
- Representing the buttons with something other than colours might be practical for buttons that already display some symbology (eg. play / forward / backwards) or have a distinctive physical arangement (eg. a D-pad). In these cases small symbols could be displayed, removing the necessity for users to memorize the button-color association.
- The corpus must be representative of the text to be entered (otherwise typing will be inefficient). The corpus could be generated using the information on the device. For example, the text-field used to search for an iPod track could use the collection of track, album and artist names as the corpus to increase the likelihood of efficient text-entry.
- Fixed letter-frequencies can be used to estimate key probabilities instead of the constantly adjusted probabilities proposed. This fixes the color sequences and makes it possible to memorize the color sequences possibly resulting in advanced typists being able to use body-memory to type without looking at the screen. The disadvantage of this approach is that the typing efficiency (as measured by the ratio of button presses to keys activated) diminishes.