January 2003
Features

Dashing along

by Helen Joyce

Cutting a dash

Eyes didn't evolve to push buttons

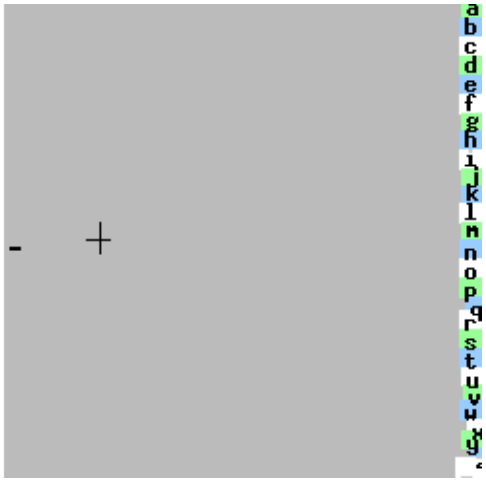
A good designer will create products that are easy to use and suited to users − but it is often forgotten just how much users adapt themselves to products, rather than the other way round. If you are a touch−typist, you

probably can't remember just how hard typing is − the awkwardness and lack of intuitive appeal of a keyboard; the way silly errors are easy to make and hard to spot. I, for example, almost always type "hte" instead of "the", and when I'm finished typing use a "find−change" to correct the error. But "hte" isn't even a proper English word, so why should it be so easy to make this mistake?

However hard typing is, at least after some practice it becomes second nature. Imagine, however, that you are disabled and cannot speak or use your hands to communicate. Many such people use some sort of eye−driven device to pick out words laboriously, by spelling them and sometimes using a list of completions − basically, using their gaze to push metaphorical buttons. The resulting text may be speech−synthesized − this is how Stephen Hawking writes and communicates, for example.

"But eyes did not evolve to push buttons", says David MacKay of the Cavendish Laboratory in Cambridge. Such a method of communication is exhausting and slow, and David reckons he and his colleague David Ward, have found a better way, which they call *Dasher*.

Pick a book − any book



Dasher awaits your commands

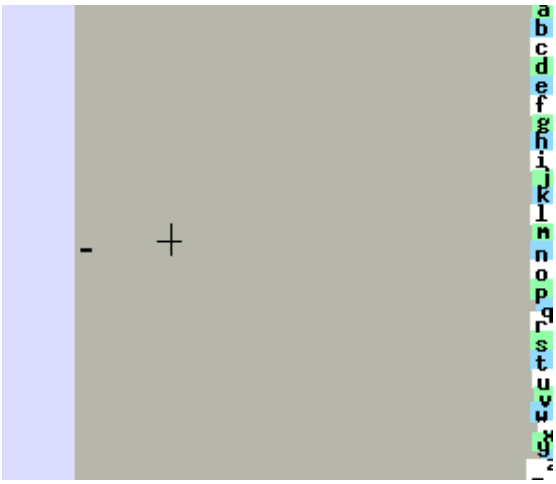
Dasher is a way of inputting data − just as typing on a keyboard is − in which the user "navigates" through all possible texts, always choosing the destination according to what she wants to say. The analogy that is often used is that of the "Library of Babel" − an imaginary library in which every possible book appears. The books are imagined to be arranged on enormous shelves in alphabetical order by content − the farthest left book just contains the letter "a", next to it is "aa", somewhere to its right is "all good things come to an end", and so on. You can imagine writing a book by "selecting" the right book from the Library of Babel − which somewhere, on its shelves, contains this exact article.

Of course this analogy feels a little strained to me sitting at my keyboard typing away − but what if, instead of typing, I somehow "zoomed in" on the text I wanted? This is exactly what Dasher feels like, and suddenly the Library seems more real.

The crucial thing is that "likely" books are given more shelfspace than unlikely ones, making the best books easy to find. I *could* write an article that started "kdhfoaute9l tekajd ld thale djfdlkj te" − but if I did that too often, *Plus* wouldn't have many readers! But just such an article sits on a shelf in the Library of Babel − however, it's written in a very small font in a very small book on a very small shelf, and noone is ever likely to find it. By contrast, sensible English sentences like "A good designer will create products that are easy to use

and suited to users" are written in normal size font and books, and easy enough to find if you go to the "A" section, and then within that section to the "A[space] " subsection, and then to "A g", and so on.

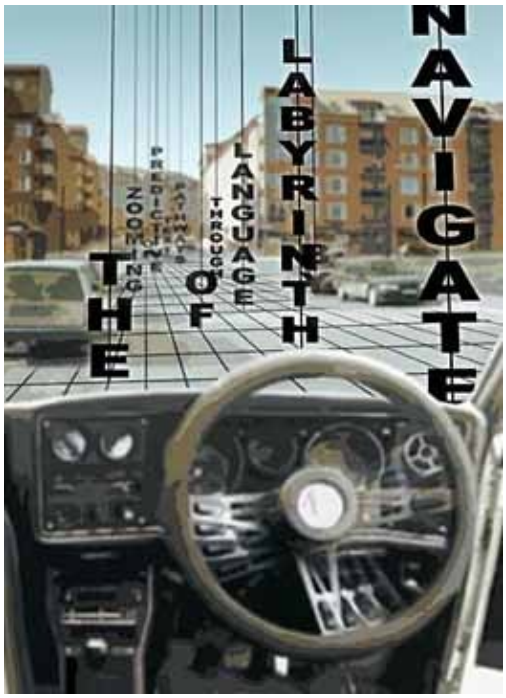
Dasher says hello



So to enter text using Dasher, you just use your mouse, or trackerball, or even your gaze, to "point", or "steer", towards the shelf in the Library of Babel where your text appears, then you narrow down some more to the "subshelf", and just keep going until you've found the book you want. That's all there is to it. It's hard to make mistakes − for example, it's hard to write my bugbear "hte" because inside the "h" shelf not much space is given for "ht" − this is a progression that doesn't appear in English. The big shelves inside "h" are "ha", "he", "hi" and so on − to the right, you can see what it would look like to use Dasher to write "hello".

That's the principle − but how does Dasher know just what is likely or unlikely? For this Dasher needs a *language model*, and the one it uses is called *prediction by partial match*. In this model the program is constantly learning from what you've already entered − it guesses what you are likely to enter next, and makes each letter easy or hard to get to accordingly, based on either the previous four letters ("lite" versions for low−memory devices) or five letters (for PCs).

This procedure is the reverse of that used by an elegant method of data compression known as *arithmetic coding*. David MacKay explains that "the idea of data compression is to take a text file, say, and compress it into a really short binary file. Arithmetic coding is a state−of−the−art way to do that sort of compression. It works by taking your text file and turning it into a string of 0s and 1s with a decimal point in front of it − it turns your file into just one real number.



Steering through the Library of Babel

"When we write text, we're making real numbers somehow − we make gestures, we're pointing, we're touching, whatever, and we want that to produce lots of text. That's the aim when you're writing, you want to write as fast as possible, so we would like to have a real motion that our hands or eyes could do, and we'd like it to turn into lots of text. That's the opposite operation, so I said 'let's make an inverse arithmetic coder, so we'll take this mapping that arithmetic coding does and turn it on its head, we'll use steering gestures, our pointing gestures, to create a real number'."

Arithmetic coding works by assigning separate parts of the interval [0,1] to each of the possible symbols a user can input. It does this according to the likelihood of each symbol. Very few articles will start with the letter "z" for example − so this letter will be assigned much less than its "fair share" of the interval − and many start with "a", so this letter will be given correspondingly more. And after each letter some letters will be more unlikely than others, so that share of the interval will also be divided up unevenly among all the possible next letters, and so on. In this way every possible text can be assigned a single, unique real number between 0 and 1, written in binary. Larger chunks of the interval are dedicated to "likely" texts, and smaller to less plausible ones.

Dasher uses the inverse idea − the line segment [0,1] is presented to the user as a long target, containing all available symbols, with larger chunks assigned to the symbols more likely to be used. At first the program guesses which symbols are more likely based on a "training text", but Dasher learns constantly, so if you are using unusual technical words or abbreviations, they will soon become as easy to input as common English words.

Who is Dasher for?

One huge advantage of Dasher is that it completely bypasses the need for a keypad. All that is needed is a device that can be used to point − a mouse, touchscreen or trackerball, for example, or even an eyetracker. This means that handheld devices with even moderately−sized screens can be used in a more natural way than is possible with a very tiny keypad − a moderate increase in the size of a mobile phone could make Dasher a

plausible replacement for irritating guessing games while texting − to the joy of SMS addicts everywhere! Palmtop computer users can already benefit, as can severely disabled users, who can use an eyetracking system to "point" at the letters they want, rather than any of the laborious "button−pushing" applications currently in use. The eyetracker effectively turns the disabled user's eye into a steering mechanism to find a way through the Library of Babel to the right text.

Using Dasher with an eyetracker

Dasher offers potential as an input mechanism for people who speak languages that are difficult to represent using a keyboard − for example, Japanese. Extra symbols can be added easily, so specialist users, such as mathematicians, might find Dasher a quick and easy way to bypass Equation Editor and its ilk, for example.

Novice users can achieve impressive speeds very quickly − and experienced users can rival handwriting speeds (25 words per minute) with an eyetracker, and around 40 words a minute with a mouse. This is not as quick as a touchtypist, but it only requires one finger, and doesn't take weeks of practice! And the best thing is − Dasher is freeware, with its creators, David MacKay and David Ward, committed to the open−source development model. You can download Dasher from their site http://www.inference.phy.cam.ac.uk/dasher/ for PCs running Windows or Linux, and for some Palmtop devices. So why not dash along, get a copy, and start Dashing?

About this article

Helen Joyce is Editor of *Plus.*

David MacKay

For this article, Helen interviewed David Mackay, a researcher on Information Systems at the Cavendish Laboratory at Cambridge. David is cofounder of the information technology company, Transversal.

David Ward, formerly a PhD student in the Physics department at the University of Cambridge, now works for a software company.

Plus is part of the family of activities in the Millennium Mathematics Project, which also includes the NRICH and MOTIVATE sites.