Bliss-Dasher – Efficient semantic writing with any muscle

David MacKay, Seb Wills, and Annalu Waller[*]

May 2007

Abstract

The full computerization of Blissymbolics is essential for its development. Full computerization means: defining a list of Unicode-compatible characters; getting that character list incorporated into the Unicode standard; and creating a free, open-source, scaleable font that software developers worldwide can use.

To demonstrate what can be done with full computerization, we have combined Blissymbolics with the information-efficient writing system, Dasher, and an open-source font provided by George Sutton.

Bliss-Dasher is an efficient and fun way of writing Bliss. This paper describes in detail how we did it.

DJCM accepts full responsibility for any errors, for example in the naming of blissymbols.

Motivation

We are interested in enhancing the accessibility of Bliss. In particular, we want to make it easy to write Bliss on a computer.

Dasher is a writing system that makes possible efficient text entry in any language composed of sequences of symbols. Dasher requires only simple steering gestures, so it is accessible to many users with physical disabilities. In English (an alphabet with 103 characters), users steering Dasher with a mouse can write at 25 words per minute after one hour's practice, and experienced users reach 35 words per minute. Dasher can also be steered by gaze-tracker; the results are record-breaking: 29 words per minute by gaze alone, for an experienced user. Dasher can also be used to write Korean (a language with 11,000 characters, each one assembled from 67 elementary 'combining' characters). Thanks to the help of language experts worldwide, Dasher now works in almost every written language. These successes convince us that we should get Bliss working in Dasher too.

[*] David MacKay and Seb Wills: Cavendish Laboratory, 19 J J Thomson Avenue, Cambridge. CB3 0HE, UK. mackay@mrao.cam.ac.uk, saw27@mrao.cam.ac.uk. Annalu Waller: School of Computing, University of Dundee, UK. awaller@computing.dundee.ac.uk.

Bliss

Blissymbolics is a graphic semantic language created by Charles K. Bliss (1897-1985) for international communication. Bliss sentences are composed of blisswords, and blisswords are composed of characters.

Thousands of Bliss characters are recognised, but most of them can be described in terms of roughly one hundred key symbols. For example, the symbol for food ⍜ is composed of the symbol for mouth ○ and the symbol for ground _ .

Similarly, the symbol for sound ʔ̲ is composed of ear ʔ and ground _. The symbol for language ʔ○ is composed of the ear ʔ and mouth ○.

Bliss is wonderfully logical: the meaning of characters can often be deduced from the component shapes or keysymbols of which the characters are composed. For example, given that father 介 is composed of man ⅄ and protection ⌃; and that the symbol for woman is ⅄̲, what do you think 介̲ means? That's right - mother. Another example: by adding to material # the symbol for protection ⌃, we obtain clothing ⌃#.

Electronic Bliss

For Bliss to be written easily on a computer, we require the following four things.

1. A **font** containing all the desired characters of the alphabet, or capable of rendering them by combining-operations. There must be a **free** font, otherwise it's impossible to make Bliss readable in all web-browsers and all text-editing software.

2. An association of all the characters with **Unicode**, the international computer-writing standard. When characters are associated with Unicode code-points, written material can be saved in ordinary files, sent by email, and rendered in web pages. Any software that is Unicode-compliant will instantly work with the language. The benefits of Unicode are nicely illustrated by Dasher: Dasher uses Unicode; as a result, Dasher worked instantly in all the character-sets of Unicode. 'Doing things right' in this way allows better accessibility. For example, users with visual impairment can choose their preferred font size and colour; searching and other text-editing operations become easy (whereas they are impossible if the characters are represented by images); and we will no longer need to worry that 'in transferring the Bliss-words from one format to another for the purposes of displaying them on the web, quality is lost.'[1]

3. An **alphabetical system** that allows the user to find and select whatever symbols she desires.

4. **Software**: software for writing Bliss efficiently, at the same speed that people can write other languages; and software that allows expert users to find all the symbols they wish to use.

[1] http://www.blissymbolics.org/moodle/

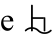
Fonts and Unicode

We are aware of some previous work to get Bliss into Unicode, and to make fonts for Bliss.

There is a spectrum of approaches. At one end of the spectrum are fonts containing only primitive Bliss strokes and spaces, from which keysymbols and compound characters could be created. For example, Douglas Crockford created a font which contains only 79 elemental Bliss strokes and spaces.[2] The idea is that software can combine these elemental strokes to create any Bliss character. This type of Bliss font is a useful tool to allow computer programs to "draw" Bliss symbols, but is not sufficient as a standard electronic representation of Bliss, because different software might combine its elements in slightly different ways (or in a different order) to create what is intended to be (and may even appear as) the same Bliss character. The raw strokes and spaces are too low-level for human-friendly interaction. We feel that, to ensure complete coverage, an international standard for Bliss should probably include these elemental strokes, but they should not form the basis of normal electronic communication in Bliss.

At the other end of the spectrum are fonts whose glyphs are entire Bliss characters, not primitive strokes. Michael Everson submitted a proposal for the inclusion of a set of Bliss characters in Unicode in 1998.[3] George Sutton's fonts, which are freely available, work in the same way. But complete coverage of all the recognised characters of Bliss makes a lot of work for font designers!

In the middle of the spectrum there could be fonts making use of combining characters to reduce the total number of glyphs required.

We believe that there should be a move to standardize a limited set of characters, never to be expanded. We feel that the Bliss community has allowed itself to create too many characters. It's unreasonable to expect a font designer or Unicode programmer to include frivolous 'characters' such as the raccoon or bear face. Indeed we feel that the inclusion of characters such as the tractor, pipe, turtle, and camel misses the point of Bliss, which is that it is a language, not a drawing system. We also feel that many of the combined characters of Bliss such as the toilet can be safely decomposed into a sequence of original characters. For example, we propose to write toilet as ⊢∼; and aeroplane as ϒ⊗ (wing, wheel). Superposing characters was easy when Bliss was written only by hand, but in computerized writing, every definition of a superposed character like  requires that a new Unicode entity and new font glyph be created. Keeping superposed characters remains an option, but keeping many superpositions will have consequences for the complexity of programming and using a computer-based writing system (especially in a keyboard-based approach). Of course, not all such superpositions of characters can be undone in the way we suggest for toilet ⊢∼: the superposition of nose ∠ and mouth ○, for example, means *taste* . But the sequence ∠○ means *breath*. So we probably can't strip Bliss right down to its keysymbols. Some compound characters must be kept. One of our aims in Bliss-Dasher will be to make it as easy as possible to select whatever compound characters are retained.

[2] http://www.crockford.com/blissym

[3] http://std.dkuug.dk/JTC1/SC2/WG2/docs/n1866.pdf

We emphasize that Bliss-Dasher will work fine even if many compound characters are retained. Indeed, the more characters Bliss has, the greater the advantage of Dasher over keyboard-based approaches to writing.

Dasher

Dasher is a text-entry interface driven by continuous two-dimensional gestures, delivered, for example, via a mouse, touch screen, or eyetracker; the user writes by steering through a continuously expanding two-dimensional world containing alternative continuations of the text, arranged alphabetically. Dasher uses a language model to predict which letters might come next and makes those letters easier to write. The language model can be trained on example documents in almost any language, and adapts to the user's language as she writes. Dasher is free software.

We now describe Dasher in its English-writing mode. Imagine writing a piece of text by going into the library that contains *all possible books*, and finding the book that contains exactly that text. In this way, writing can be turned into a navigational task. What is written is determined by where the user goes. In Dasher's idealized library, the 'books' are arranged alphabetically on one enormous shelf. When the user points at a part of the shelf, the view zooms in continuously on that part of the shelf. To write a message that begins 'hello', one first steers towards the section of the shelf marked h, where all the books beginning with h are found. Within this section are sections for books beginning ha, hb, hc, etc.; one enters the he section, then the hel section within it, and so forth.

To make the writing process efficient we use a *language model*, which predicts the probability of each letter's occurring in a given context, to allocate the shelf-space for each letter of the alphabet, as illustrated in figure 1. When the language model's predictions are accurate, many successive characters can be selected by a single gesture.

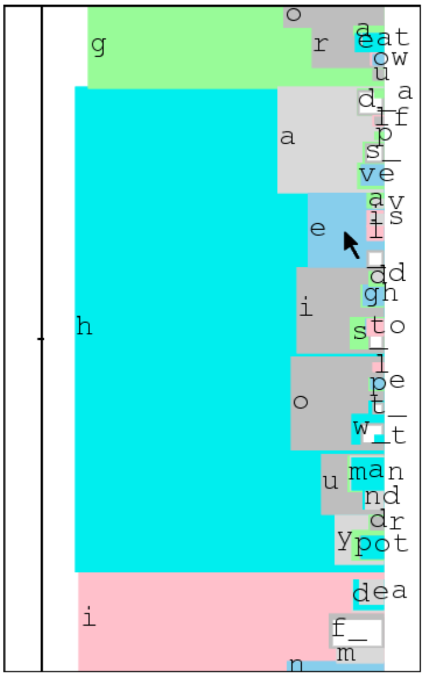


Figure 1.

A screenshot of Dasher's canvas when the user starts writing hello. The shelf of the alphabetical 'library' is displayed vertically. The space character, '_', is included in the alphabet after z. Here, the user has zoomed in on the portion of the shelf containing messages beginning with g, h, and i. Following the letter h, the language model makes the letters a, e, i, o, u, and y easier to write by giving them more space. Common words such as had and have are visible.

The pointer's vertical coordinate controls the point that is zoomed in on, and its horizontal coordinate controls the rate of zooming; pointing to the left makes the view zoom out, allowing the correction of recent errors.

With Dasher, it is easy to spell correctly and hard to make spelling mistakes. You can download Dasher (for Linux, Mac, or Microsoft Windows) from `http://www.dasher.org.uk/`. There is also a Java version that runs in your web browser.

Making Bliss-Dasher

The font, and a temporary home in Unicode

To make the first version of Bliss-Dasher, we used free fonts kindly provided by George F. Sutton at Symbols.Net (`http://www.symbols.net/zips/`). These fonts provide hundreds of Blissymbols, including many compound symbols. The fonts lacked a few keysymbols, so we added a couple of glyphs by hand, making a new single font file that we named SuperSemantic.ttf. We mapped all the glyphs into the private-use area of Unicode. We then selected 245 keysymbols and 260 compound symbols from these fonts, for the first Bliss-Dasher prototype.

For Dasher, all the characters must be arranged in an ordered list. The list may be subdivided into groups, subgroups, and so forth, to any depth. We considered using shapes as the organizing principle; we also considered defining an alphabetical order by semantic meaning. The Bliss alphabet song and the Blissymbol Reference Guide organize symbols by shape using the following groups (reading down the first column, then the second).

Wavy lines
Heart
Cross hatches
Building
Ear
Arrows
Wheel
Large circle
Small circle
Half circles
Quarter circles
Parentheses
Squares
Rectangles

Open squares
Open rectangles
Right triangles
Dot
Right angles
Lines on a base
Crosses
Isosceles triangles
Symmetric acute angles
Asymmetric acute angles
Horizontal lines
Vertical lines
Slanted lines
Diagonal lines
Alphanumeric characters

In a strictly shape-based system, presumably we would go within the ‘small circle’ group to find flower ⚲, colour ⍜, food ⍜, girl ♀, and boy 웃. From a semantic point of view, however, there’s no relationship between these five entities, except for girl and boy. Girl and boy are both semantically much closer to man and woman than to any of flower, colour, and food. The shape-based system could instead file girl under Isosceles triangles, and boy under Symmetric acute angles. This would put the girl close to woman and the boy close to man, and would get them both away from the food; but it would separate them from each other, and from baby and teenager.

We are not dogmatic about the best way to organize blissymbols. What we have used for our first demonstration is a compromise order - part semantic, part shape-based - that we, as users, found easy to master. We emphasize that Dasher is a flexible system, compatible with any order. So if you don’t like our order, feel free to substitute your own. Dasher is free software, and the alphabetical order is easy to edit.

Our alphabetical order

We define the alphabetical order in terms of groups and symbols. For example, the alphabetical order for English consists of two groups:

group 1	
name:	“lower case roman letters”
label:	“”
symbols:	a b c d e . . . z
group 2	
name:	“punctuation characters”
label:	“”
symbols:	! ? , _

Each group contains an ordered list of characters. Groups may contain groups within them, to any depth. We use the terms *parent* and *child* to describe the relationship between a group and the characters and groups that it contains. We call a pair of symbols or groups ‘siblings’ if they have the same parent.

Each group has a *name*, and an optional *label*. The *name* is a description of the group (like ‘wavy lines’); the *label* is an optional symbolic abbreviation for the group that may be used in user interfaces (just as the symbol ≀ (fire) is used as a key symbol for the group of wavy lines in the Blissymbol Reference Guide). We emphasize that a group is not itself a character, even though we may label the group by a character. Every desired symbol of the alphabet must appear explicitly as a character once in the alphabet. For example, if we have a ‘wavy lines’ group whose label is ≀ (fire), we will include the character ≀ (fire) in that group. We might say that ≀ (fire) the character is a child of ≀ (fire) the group.

We will use groups to group semantically related characters such as girl and boy. We will also use groups to represent the descent of compound characters from their parent key symbols. For example, food is descended from mouth ○ so we will make a group named mouth, and labelled by ○, which will contain within it (food). The group named mouth will also contain the character ○ (mouth). We usually put such a character first in the group.

Our alphabet has 507 characters and 229 groups. We'll describe the whole alphabet in due course, but first we will summarise the top-level groups by naming them, giving a few examples of symbols found within each group, and making a few comments on whether each group is principally shape-based or semantics-based.

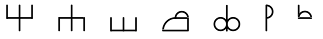
Please make sure to read the whole summary before trying to find a character. We emphasize the alphabet is not solely organized by shape, so two very similar-shaped symbols may not be in the same group as each other.

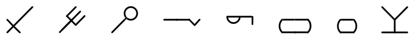
Group name	**Description**
Elements ≀	The first group contains the wavy lines (fire ≀, water ∼) and electricity ϟ and electromagnetic radiation ⏦. The label for group 1 is fire ≀.
Lines and measures ☰	The second group contains sky ‾ and earth _ and other simple horizontal and vertical lines, including several measures of size (tall I, short ɪ, wide ⊢⊣, narrow н). The temperature symbol ‡ also belongs here, as does 'number' #. The 'elements' sky ‾, earth _, air ⅂, and wind ⅂→ are all in this group, so the 'elements' theme blends over from group 1. Maybe group 1 should be called 'wavy and jagged elements', and group 2 'straight elements'. Less obvious residents here are cloth ⌗, structure ⁞⁞, joint ⁞_, and bone ⁞, all of which are principally composed of horizontal and vertical lines. The label for group 2 is 'sky-mid-earth' ☰.
Sun-time ○	Continuing the theme of measurement, the third group contains all the planetary symbols relating to times ◷ - sun ○, month D, star *, night ⟫; also life ⦶ and being ⊕. The label for group 3 is sun ○.
Body ∟	The fourth group contains all symbols relating to the human body, from head to toe. In addition to symbols like eye ⊙, mouth ○, and torso ▢, we also include 'standing body' ∟ and weak ∑. The label for group 4 is body ∟.
People ⊥	The fifth group contains person ⊥, need ⟋, support ⊿, and all characters like man ⅄, daughter ⌂, and baby ⊶. We put action-man ⅄ before creation-woman ⅄, by the way, because one of our organizing principles is to put symbols with fewer pen-strokes before symbols with more. The label for group 5 is person ⊥.

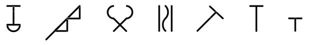
Angles ⌃	The sixth group contains acute angles such as action ∧ and ability ∨, right-angles such as protection ⌃, and also bent ⎺⟍ and broken −⟍. We order acute angles before right-angles. We order up-pointing angles ∧ and down-pointing angles ∨ before left-pointing angles < and right-pointing angles >. The label for group 6 is protection ⌃.
Triangles △	The seventh group contains creation △, make ▵, and mountain ⊿. We order isosceles triangles ▵ before right-angled triangles ⊿, and up-pointing objects △ before left- and right-pointing triangles ◃ ▹. The label for group 7 is creation △.
Squares □	The eighth group includes many symbols to do with enclosures and furniture, such as thing ▫, closed □, open ⊔, chair h, and stair ┌┌. Most of the symbols in this group have only horizontal and vertical lines, meeting at right-angle corners. The key symbols are ordered thus: complete squares (large then small); fragments of squares; complete rectangles (upright then horizontal); furniture and miscellaneous cornerish left-overs. The label for group 8 is closed □.
Half circles ◠	The ninth group contains the mind ◠ and the bowl ◡ (but not the past); we already put that in the sun-time group). We also put knowledge ∩ in here, thanks to its derivation from mind ◠. The order of the key symbols is 'up before down': ◠ ∩ _◠_ ◡ ‾◡‾ - mind, knowledge, bridge, bowl, and hole. The label for group 9 is mind ◠.
Pictograms ⟍	What symbols in the pictograms group have in common is that none of them has a clear semantic connection to any simple keysymbol. For example, scissors ȸ contains two circles but has no semantic connection to mouth ○. This is a big group and it has several important subgroups. We arrange the pictograms semantically in the order: 10a: human things (eg, boat ⍦, needle ᑭ, cut ⤫, wheel ⊗, chemistry ბ), 10b: greenery (leaf ⦶, tree 个), 10c: animals (heart ♡, wing ϒ, fish ∝), 10d: abstract symbols such as combine ⚭ and blissymbol ∑. The label for group 10 is pen ⟍.
Arrows →	The eleventh group contains, for example, forward →, back ←, jump ↷ - all the basic arrows. The order is up, down, left, right, diagonal, round. The label for group 11 is forward →.

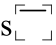
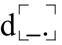
<table>
<tr><td>Prepositions
·|</td><td>Examples in group twelve are: before ·|, over ∸, under ⨪, into ⍈, through ⇸, for > and against <. The prepositions are sorted by shape. The label for group 12 is before ·| (since ‘before’ is the meaning of the word ‘preposition’).</td></tr>
<tr><td>Little words
⊣</td><td>Group 13 contains the ∕, this ∕˙, that ·∕, a ∖, it ı, and but ⊣. The label for group 13 is ‘but’ ⊣.</td></tr>
<tr><td>Modifiers
+</td><td>Group 14 includes punctuation, strategies, dots, indicators, and markers. The label for group 14 is plus +.
Group 14’s subgroups are as follows:
<table>
<tr><td>14a: punctuation</td><td>quote and comma. (In accordance with Dasher tradition, we put newline and space at the very end of the alphabet.)</td></tr>
<tr><td>14b: strategies</td><td>meaning-modifiers such as delete ∕, opposite ↿, very !, generalization ♀, metaphor ⫯, past), and present)(. We put negative things first (delete ∕, opposite ↿, minus —), then plus +, many ×, part ÷, and equal =; then metaphor ⫯; then the alphabet-like entities, very !, query ?, past) and future (.</td></tr>
<tr><td>14c: dots</td><td>Dots are ordered from top to bottom.</td></tr>
<tr><td>14d: indicators</td><td>Past indicators, verb and adjective indicators.</td></tr>
<tr><td>14e: markers</td><td>We currently have three markers for left and four for right.</td></tr>
</table></td></tr>
<tr><td>Numerals</td><td>Group 15 contains 1, 2, 3, ... , 9, 0</td></tr>
<tr><td>Roman</td><td>Group 16 (optional) contains a...z.</td></tr>
</table>

The pictogram subgroup 10a called ‘human things’ is further subdivided into subsubgroups: ‘pastimes’, ‘home’, and ‘work’ (eg, flag , wheel ⊗, money , medicine , chemistry). The subsubgroup ‘home’ is further subdivided into ‘clean’, ‘dine’, and ‘toolbox’.

Clean: 

Dine: 

Toolbox: 

When we display the characters on the Dasher canvas, we try to avoid confusion among similar characters by surrounding appropriate characters with a ‘combining fiducial glyph’; For example, we present sky ‾ and earth _ as  and 

Within each group we use the following organizing principles to order the characters:

1. We tend to progress from top to bottom;

   from large to small;
   from a basic object to derived objects; and
   from fewer strokes to more strokes.

2. Where multiple objects are derived from one character, we sort them by shape, using the same shape-sequence as the shape-sequence of our top-level groups:

   wavy lines,
   horizontal and vertical lines,
   circles,
   angles,
   triangles,
   squares,
   half-circles,
   arrows,
   pictograms (including the pen diagonal),
   arrows,
   cross-out diagonal,
   and numerals.

Colouring

We colour every group and every character. (Again, like the alphabetical order, the colour scheme is easy to alter; it’s defined in a couple of XML files.)

The following principles apply. The first ten top-level groups are coloured with rainbow colours, and the remaining top-level groups with four other colours as follows:

Group name	Colour
Elements ≀	Red
Lines and measures ☰	Orange
Sun-time ○	Yellow
Body ∟	Lime
People ⊥	Green
Angles ^	Teal
Triangles △	Light blue
Squares □	Blue
Half circles ◠	Purple

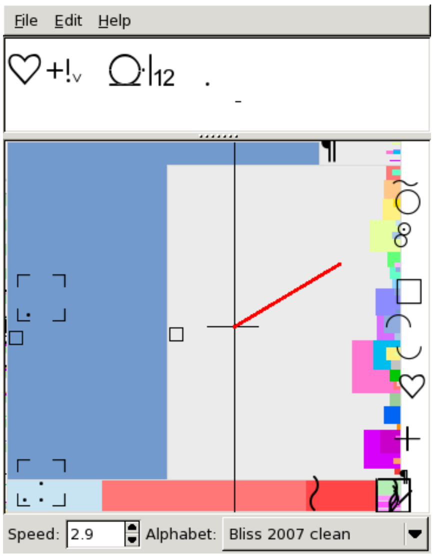
Group name	Colour
Pictograms ╲	Intense pink
Arrows →	Khaki
Prepositions ·|	Dark blue
Little words ⊣	Brown
Modifiers +	Dark purple
Numerals	Red
Roman	No colour
Paragraph, space	White

We use shades of magenta, slate-blue, and salmon to colour characters. Where a character appears twice, once as the label of a *group*, and once as the *symbol* itself, we colour the box of the *symbol* with a distinctive milky green colour, the same as the starting colour of the Dasher canvas. This is our "go" colour, our "let's **go** and get the next character now" colour. We colour all such symbol-boxes '**go'** green to distinguish the act of "choosing the character that owns the current group" from the alternative, "choosing one of the derived characters".

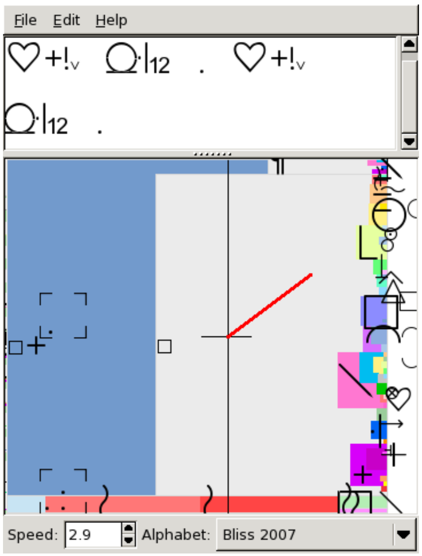
We've made several other symbol-colouring decisions, which we hope the user will pick up along the way. For example, within the 'greenery' group, we not only order the objects of the plant kingdom from top to bottom (flower ♀ fruit ϭ banana ʆ tree ↑ leaf Ø grass ⌐ rice ꭓ root ⊼ and seed ⩪); we also colour them memorably: flower ♀ is red; fruit ϭ is orange; banana ʆ is yellow; tree ↑, leaf Ø, and grass ⌐ are shades of green; rice ꭓ is blue; and having got so far in rainbow order, we felt compelled to make root ⊼ and seed ⩪ deep blue.

Group labels

We've made two versions of this alphabet. In the alphabet called "**Bliss 2007**", top-level groups and some intermediate groups are labelled, on the Dasher canvas, by their group labels. These labels may help the user find their way down the alphabet tree to their next character. On the other hand, having lots of labels in the tree means that the sequence of characters seen in the Dasher canvas is much longer than the text that is written - at least twice as long, since every character is preceded on the canvas by the label for its top-level group. The labels may prove distracting or confusing. So we have a second version of the alphabet called "**Bliss 2007 clean"**, in which every available character in the alphabet is rendered exactly once in the alphabet tree; there are no top-level labels. The colour scheme remains identical to that in "Bliss 2007", so a user who becomes familiar with Bliss-Dasher using "Bliss 2007" may find it easy to switch to "clean" in due course.



Speed: 2.9 Alphabet: Bliss 2007 clean



Speed: 2.9 Alphabet: Bliss 2007

Figure 2: Screenshot when using “Bliss 2007 clean”. The Dasher application has a ‘text box’ at the top, displaying what has been written; and the ‘Dasher canvas’ below, where the zooming happens.

Screenshot when using “Bliss 2007”

Also, for explanatory purposes, we made a stripped-down alphabet called “Bliss 2007 keysymbols”, which contains only the characters that we view as fundamental, and none of the more complex compound characters.

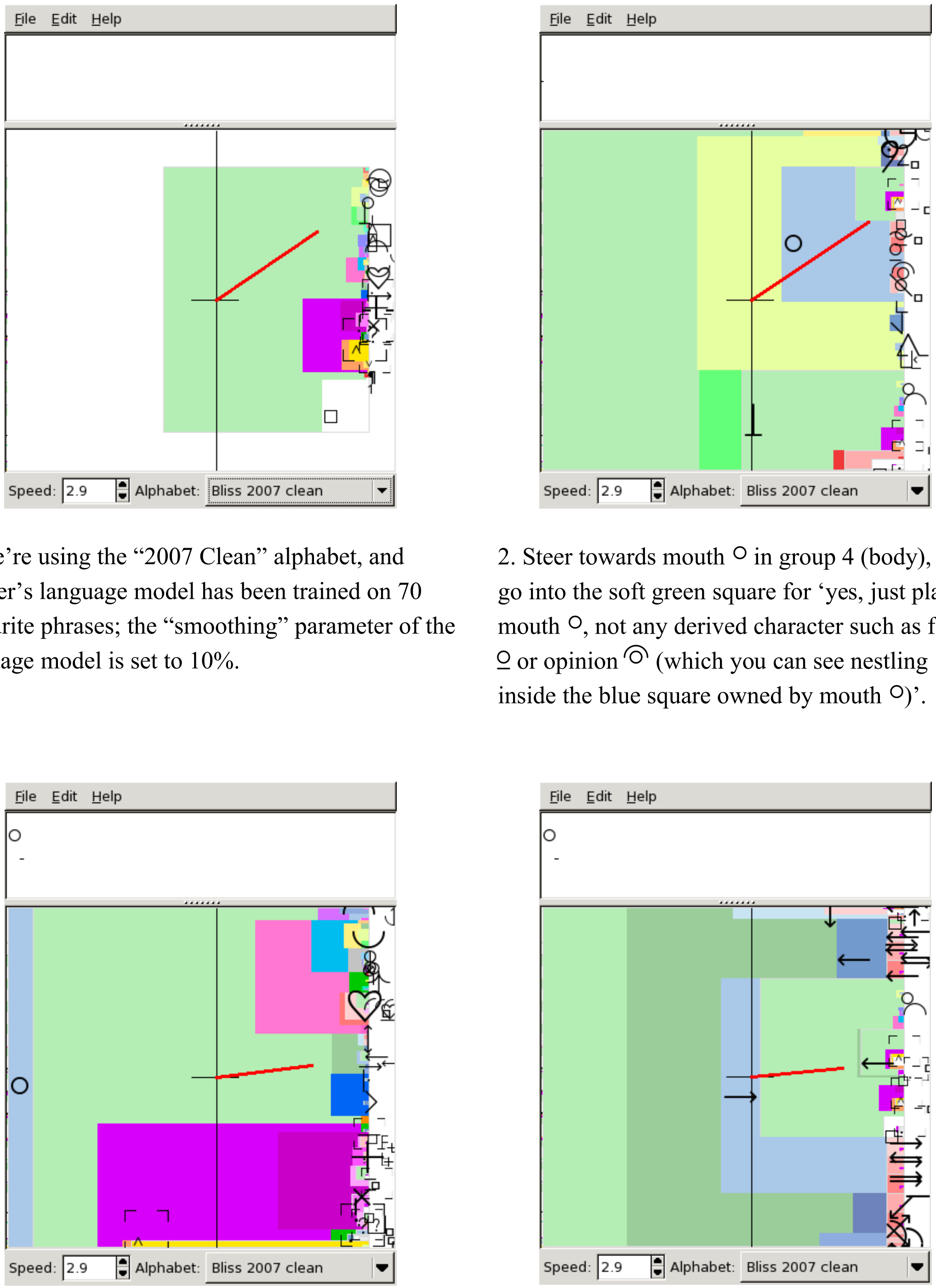
Using Bliss-Dasher

Selecting a desired character from an alphabet of 500 sounds daunting, but Dasher makes it easy, assuming the user knows the semantic route to the character. If Dasher has not been provided with any training text, so that it assigns all characters equal probability, an able-bodied user (DJCM) writes Bliss at a speed of roughly 1 character per 4.5 seconds, steering with a mouse. With an appropriate training text, writing with Dasher gets faster, as the language model learns which are the probable characters and sequences of characters.

When Dasher has been trained with a training set consisting of ‘70 favourite phrases’ (listed in the appendix), the same user needs just 4 seconds to write a 12- or 14-character phrase in the training set such as ‘Good Morning.’ ♡+!ᵥ ○·|12 . or ‘Hello, How Are You?’ ○→← , ?ᴧ ⊥2 .

The screenshots in figure 3 follow the writing of ‘Hello, How Are You?’.

Figure 3: Writing ‘Hello, how are you?’ ○→← , ?^ ⊥2 . in Bliss-Dasher.



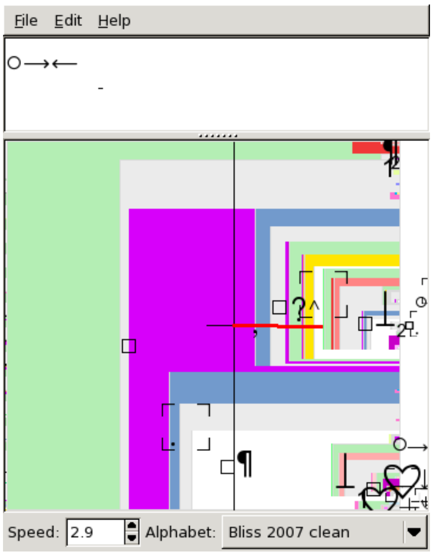
Speed: 2.9 Alphabet: Bliss 2007 clean

1. We’re using the “2007 Clean” alphabet, and Dasher’s language model has been trained on 70 favourite phrases; the “smoothing” parameter of the language model is set to 10%.

2. Steer towards mouth ○ in group 4 (body), and go into the soft green square for ‘yes, just plain mouth ○, not any derived character such as food ⍜ or opinion ⌒ (which you can see nestling inside the blue square owned by mouth ○)’.

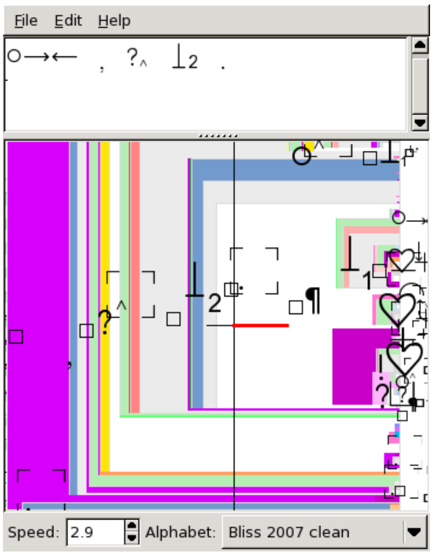
3. Now go into the khaki box where all the arrows live.

4. Choose ‘forward’ → and steer into its soft-green square to indicate ‘yes, just forward, not other derived options like race ⇉ ’. Within that soft-green box, you can already see the next character we want ←.



Speed: 2.9 Alphabet: Bliss 2007 clean

5. Go into the soft-green box owned by ←, then enter the space character (white box), then within that box find the punctuation characters and enter the comma's box. You can already see the end of the phrase coming up.



Speed: 2.9 Alphabet: Bliss 2007 clean

6. The next seven characters, ?^ ⊥2 ., are guzzled up in a single gesture.

Discussion

Using the Unicode Private Use Area allowed us to quickly build a first version of Bliss-Dasher. In the long run, official inclusion of Bliss into Unicode would be desirable to allow inter-operability and standardisation on a universally-supported set of blissymbols across all software, and to avoid clashing with other uses of the Private Use Area.

We have presented our chosen alphabetical order in detail not because we believe it is the right order, but rather because we wanted to make a working demonstration of the Bliss-Dasher concept, so we had to choose an alphabetical order; and because we'd like others to be able to try out Bliss-Dasher, so we need a document that explains our design. Feel free to change it! Web-pages for the Bliss-Dasher project are at `http://www.inference.phy.cam.ac.uk/dasher/development/bliss/` and there's a wiki too: `http://www.inference.phy.cam.ac.uk/cgi-bin/wiki/wiki.pl/BlissDasher`.

(These two URLs may be reached via `http://tinyurl.com/yvu8e8` and `http://tinyurl.com/2ho6nk`.)

We are aware of minor defects in the SuperSemantic font, but we hope these won't detract from your enjoyment of Bliss-Dasher.

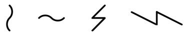
Acknowledgments

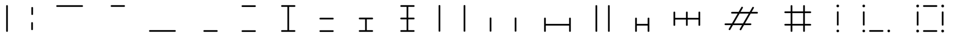
We are grateful to George Sutton, Douglas Crockford, Michael Everson, and Ross Summerfield for their open-source work on the computerization of Bliss.

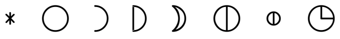
DJCM and SAW were supported by the Gatsby Charitable Foundation.

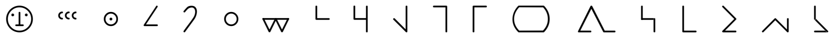
Appendix

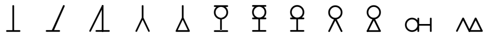
Here are the keysymbols of our alphabet, in order. (The compound characters of the alphabet are omitted.)

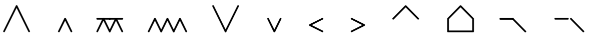
Group 1: 

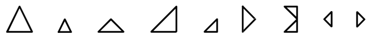
Group 2: 

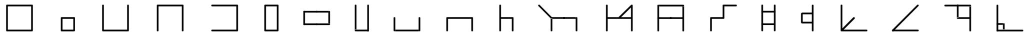
Group 3: 

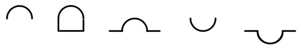
Group 4: 

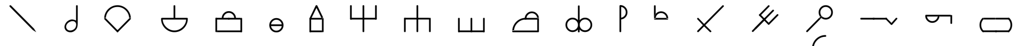
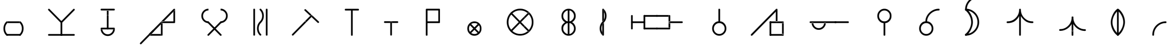
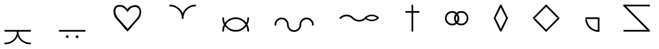
Group 5: 

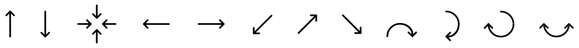
Group 6: 

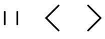
Group 7: 

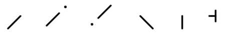
Group 8: 

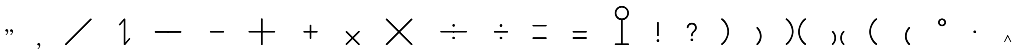
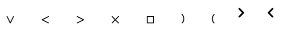
Group 9: 

Group 10: 



Group 11: 

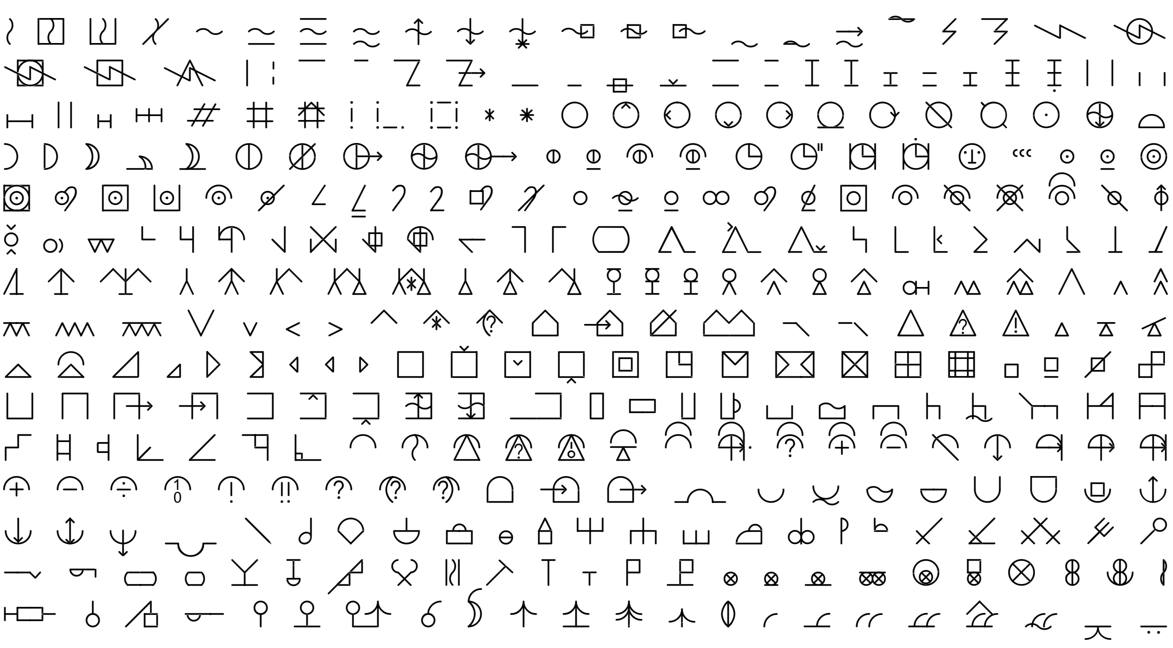
Group 12: 

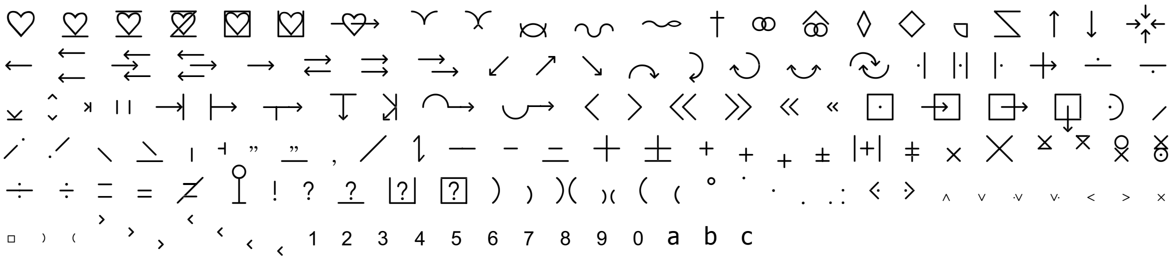
Group 13: 

Group 14: 


Group 15: 1 2 3 4 5 6 7 8 9 0

The complete alphabet in the first version of Bliss-Dasher:





Example training phrases

Hello ○→← . *(Note, we chose to use two separate glyphs to create →←; we omitted the single combined glyph from our alphabet.)*

Good Morning ♡+!v ◯·|12 .

Good Evening ♡+!v ☽ .

Good Afternoon ♡+!v ◯|·12 .

Good Night ♡+!v ☽ .

Where do you live? ?̲ ⊥2 ⌂^ .

Whats Your Name? [?] ⊥2+ ⦰ .

Pleased To Meet You ♡↑v →←^ ⊥2 .

Hello How Are You ○→← , ?^ ⊥2 .

Goodbye ○→← .

Please Play Music !♡ ^d^ .

Please Turn It Up !♡ ▷^ 2 ↑ .

Please Turn It Off !♡ ^☒^ ?d .

Please Turn It Down !♡ ▷^ 2 ↓ .

Please Scratch Me !♡ √⇄^ ⊥1 .

Please Help Me !♡ ⩘^ ⊥1 .

Please Give Me Something To Eat !♡ ⚓^ ⊥1 \□ ≫ ♀̲ .

Please Give Me Something To Drink *('drink' symbol is missing from font)*
!♡ ⚓^ ⊥1 \□ ≫ ⊜ .

Please Give Me My Book *(used 'page-page' instead of book)*
!♡ ⚓^ ⊥1 ⊥+ ▯▯ .

Please Give Me My Book *(used 'many pages')* !♡ ⚓^ ⊥1 ⊥+ ×▯ .

Pardon? ?○^ .

OK ♡+! .

No Thank You −‼ ♡⚓ .

No −‼

Its Important	ı ⦶^ ⌒!!v .
I’m Vegetarian	⊥1 −! o̲^ o̲⩚ .
I’m Tired	⊥1 ♡̲/̲^· .
I’m Thirsty *(drink symbol is missing)*	⊥1 ♡−⊖̲ .
I’m Sorry	⊥1 ♡↓⟩) .
I’m Sad	⊥1 ♡↓^ .
I’m Not Comfortable	⊥1 −! ♡̲v .
I’m Lost	⊥1 ⦶^ ↿⊙⊟v· .
I’m Hungry	⊥1 ♡−o̲v .
I’m Happy	⊥1 ♡↑^ .
I’m Fine Thank You	⊥1 ♡+!^, ♡⚓ .
I’d Like To Sleep	⊥1 ♡+!^ ⊡^ .
I’d Like To Make A Phone Call	⊥1 ♡+!(ᘓϟ^ .
I’d Like Something To Eat	⊥1 ♡+!(⟍□ ⟫ o̲ .
I’d Like Something To Drink *(drink symbol is missing)*	⊥1 ♡+!(⟍□ ⟫ ⊖̲ .
I’d Like Some Water	⊥1 ♡+!(÷v ∼ .
I’d Like Some Juice *(drink symbol is missing)*	⊥1 ♡+!(÷v ⊖̲6 .
I’d Like A Taxi	⊥1 ♡+!(⊗4◷ .
I Need The Toilet	⊥1 /̲^ ⊓∼ .
I Don’t Understand *(missing symbol?)*	⊥1 −! →⌒□^ .
I Don’t Like That	⊥1 −! ♡+!v .⁄ .
Have A Good Trip	±^ ♡+!v ⊗̲ .
Happy Birthday	♡↑v ○̲✱ .
Excuse Me Please	⍑|·^ ⊥1 !♡ .
Excuse Me	⍑|·^ ⊥1 .
Congratulations	!♡+^ .
Can You Tell Me The Time	⌊?⌋ ·v^ ⊥2 ○^ ⊥1 ◷ .
Yes	+!!
Where Is The Toilet	?̲ ⊓∼ .
Thank You Very Much	××♡⚓ .
Thank You	♡⚓ ⊥2 .

Thank You	♡⚓ .
Sorry (missing symbol?)	♡↓>) .
You’re Welcome	⚲⊥2 ♡→⌂ .
Yes Please	+‼ !♡ .
How Much Is This	⍰ 8< ⁄˙ .
How Much Do I Owe You	?× 8 ⊥2 ⚓◷) ⊥1 .
I Have A Pain Here *(missing symbol?)*	⊥1 ±^ ♡⋀ >¨ .
I Have A Headache	⊥1 ±^ ♡⋀⊤→ ⊥⃝ .
I’d Love To, Thank You	⊥1 →♡→(, ♡⚓ .
I Need A Doctor	⊥1 ∠^ ⊥≀ .
I’m Hot *(the symbols < and > are not at the right height)*	⊥1 <≀>v .
I’m Cold	⊥1 <⊡>v .
What’s Your Telephone Number	⍰ ⊥2+ ໑ϟ .
I’m Not Feeling Well	⊥1 −! ♡^ ♡+!v .

Version 1.3.9