The road to Noto

Steven Matteson

Editor's note: This is a lightly edited transcript of the talk given at the TUG 2020 conference. Some of the illustrations are omitted here; for the full set, and the video of the talk, see `tug.org/tug2020`.

The Noto family of fonts is one of the largest undertakings in the history of type founding. It certainly has not been a straight line from point A to B. I've been involved on and off for 14 years, and there are about 60 others who have contributed to it up to this day. This doesn't include the efforts on the Chinese, Japanese and Korean fonts, which people from Adobe would have to tell you about.

For the purposes of this talk the road to Noto begins with the Rosetta Stone:

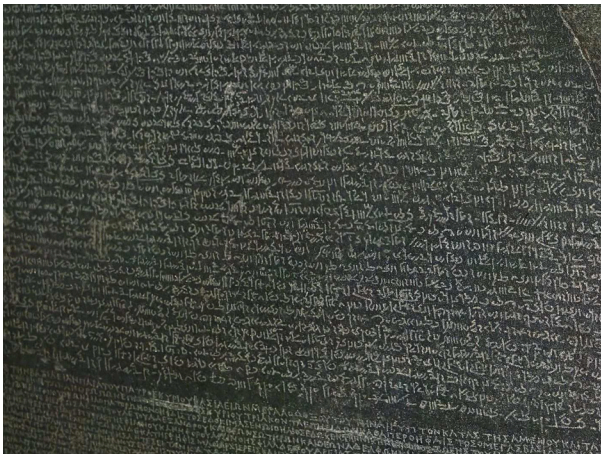

Figure 1: The Rosetta Stone, 196 BCE.

a 3.5-foot tablet fragment, similar to granite, with the remarkable workings of hand and chisel, carefully spelling out an imperial decree in three different writing systems. The stone's historical significance is legendary. I clearly remember studying it in seventh grade World History. We had to make our own version out of modeling clay and mark it with our own messages with a toothpick. We even had a contest to try and translate each other's messages. Thank you Oak Park, Illinois Public Schools.

The markings are King Ptolemy V's "Memphis Decree", given in 196 BCE during turbulent political times and cultural upheaval. The Rosetta Stone is a fascinating example of the painstaking efforts made to produce a document in multiple languages — in this case Hieroglyphs, Demotic Script and Ancient Greek — all with the aim that multiple cultures and generations understand this single message.

Fast forward to 1573 to Christophe Plantin's Polyglot Bible printed in Antwerp and funded by King Philip II of Spain.

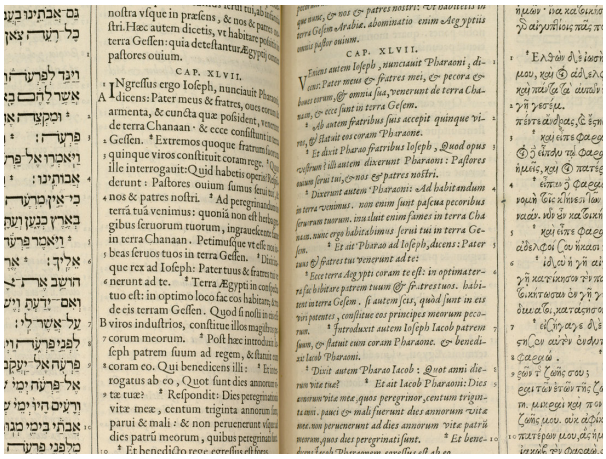

Figure 2: Christophe Plantin's polyglot Bible, 1573.

It was printed in six volumes of the different books of scripture and two additional volumes which contained translation dictionaries to accompany them. The text is translated into Hebrew, Greek, Aramaic, Syriac and Latin texts. The typography is stunning in its beauty, simplicity and painstaking planning.

The spread above shows Hebrew script on the far left with Latin in an upright roman typeface. The far right is a beautiful flowing cursive-looking Greek text; the accompanying Latin is in italic to complement the look and texture of the Greek. This typographic detail helps unify and bring harmony to the page despite the differences in the multilingual writing systems.

For me the achievement here, the complexity of the formatting and quality of printing, is inconceivably beautiful.

Starting around 1654, about 70 years later, in England, Bishop Brian Walton began work on his polyglot bible (next page). He published *nine* translations — Aramaic, Hebrew, Syriac, Arabic, Samaritan, Ethiopic, Greek and Latin. This production was funded by subscription rather than a grant from a government or church body. In just a year the bishop had found 400 private contributors anticipating the finished book.

Some consider this the least beautiful of all the polyglot bibles. It may be because it was not a royally funded project with commensurate royal flourish. Or maybe because of the complexity of nine translations vs. the four or five of previous works, which challenges the typographer to assemble a harmonized page.

Figure 3: Bishop Brian Walton's polyglot bible, 1654.

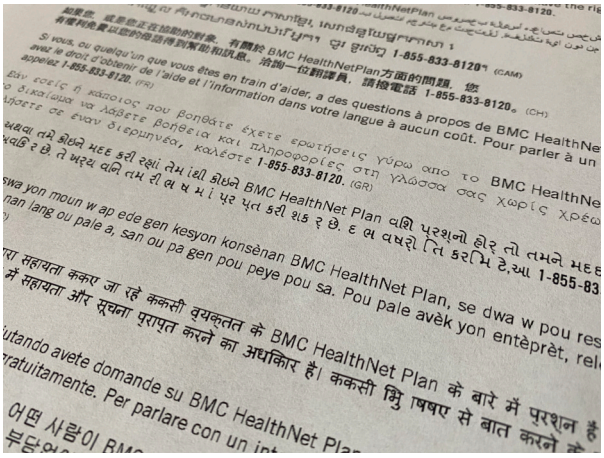

Figure 4: A common contemporary polyglot page.

But for me, Walton's achievement, the complexity of the formatting and quality of printing, is inconceivably beautiful. Particularly compared to where we are 400 years later with a standard insurance company's explanation of benefits statement (above).

This polyglot page is now commonplace and easy to do with our current typesetting tools. However, like many generic or institutional forms we see day to day, this page can be vastly improved upon. The line lengths are excessive for most of the text represented here. The variety of type styles makes it appear to be a ransom note rather than a serious document. The boldness of some of the translations makes them appear far more important than other languages, so not very egalitarian. And, from a branding standpoint, the visual identity of this company is not maintained.

I don't want to negate the complexity happening behind the scenes to make this page possible. It is far more intense than the reader will ever, or should ever, know or worry about. Just the ability to shape text right to left was a big step in computing, let alone the other magic going on here to typeset in all these languages.

Multilingual, or polyglot, typographic pages can get worse than this. An author's worst nightmare might be for his or her reader to come upon a page that looks like this:

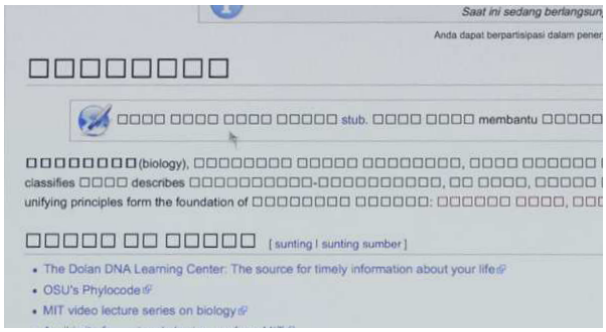

Figure 5: Tofu.

This page is full of missing glyphs. I received this image from Google's Bob Jung, who orchestrated much of the early part of the Noto project on Google's side. If a computer system is missing a character that was entered by the author, the reader sees the undefined glyph — typically an empty square. As Bob told me, "The squares remind a lot of people, particularly in Asia, of packaged bean curd. Tofu."

And that's where the Noto fonts got their name: No To(fu) = Noto.

My road to Noto began in 1985 when I started at RIT's school of printing. In my typography classes I was introduced to hot metal typesetting juxtaposed with the latest computer typesetting equipment of the day, i.e., bitmap fonts.

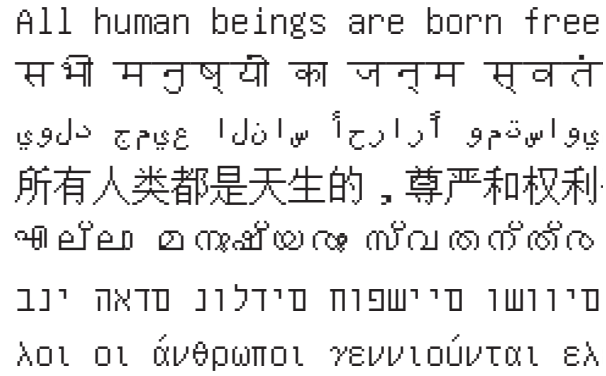

Figure 6: My road to Noto starts, 1985.

Prof. Archie Provan was working as a consultant to Xerox on their efforts to gather bitmap fonts from many different foundries that would be able to typeset all the world's languages. He was working

with Ed Smura on the AFII standard — the Association for Font Information Interchange. This included other aspects related to typography, including typeface classifications, but the bitmap font project was probably the most ambitious aspect.

With these bitmap fonts, the Xerox Star publishing system, a precursor to the Macintosh, was to be able to produce documents in any language. The pipe dream was for the fonts to be beamed via satellite to a Xerox Star installed anywhere in the world.

When describing the project to me, Archie gave a romantic notion about how, if people around the world could communicate more accurately and easily, they might spend less time fighting.

Homely as they were, these monochrome bitmap shapes would carry the power of our own messages, our own 'Memphis decrees', all around the world.

By the time I graduated, outline vector fonts that we use today were becoming more viable. Single-sized jaggy bitmap shapes were suddenly no longer sufficient when you could infinitely scale a letterform for more expressive typography.

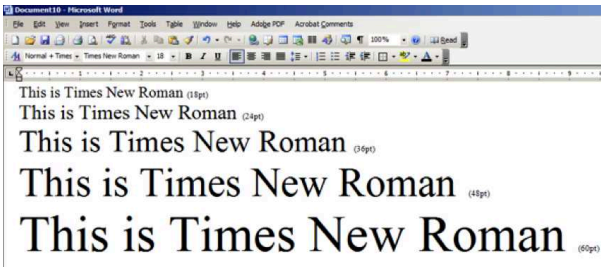

Figure 7: Early outline fonts.

I began working on the TrueType system fonts for Microsoft in 1990. The fonts all had a modest character set of around 300 characters, already more than earlier font formats could handle. TrueType fonts could handle more characters, and thereby typeset more languages, because Unicode became the standard way of encoding or 'organizing' all the letters in all the alphabets in the world, allowing (at that time) 65,536 characters in all. The previous encoding schemes, including such as ISO 8859, typically allowed only 256 characters in a single file.

With Unicode every character in the world gets a unique identifier. For example, the G-breve for Turkish gets a name and a Unicode number (Ğ, U+011E); as does the Greek Omega (Ω, U+03A9), etc. The Unicode registry is constantly being updated. In 1999, for example, Unicode consortium scrambled to put the Euro symbol (€, U+20AC) into its directory so font foundries like Monotype could update their massive font libraries to support the new currency.

Between 1990 and 93, foundries continued developing a steady stream of Unicode-encoded fonts, ever growing in size. The WGL (or Windows Glyph List) character set, defined by Microsoft, raising expectations for fonts to having support for about 90 languages with around 600 unique characters in a font file.

Bigelow and Holmes built their Lucida Sans Unicode font to coincide with the publication of Unicode's 1.0 specification. Lucida Sans Unicode was released by Microsoft in 1993 and added Greek, Cyrillic and Hebrew to the Latin Character set. The font also included support for phonetic and math symbols which Unicode had defined for version 1.0.

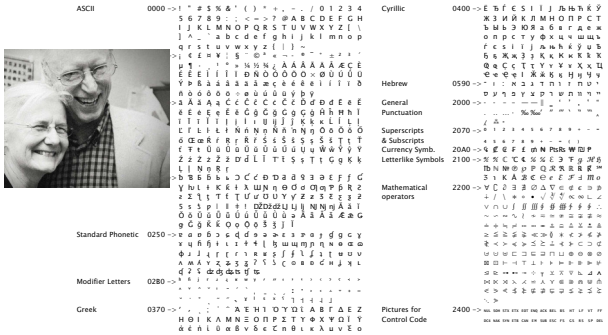

Figure 8: Bigelow&Holmes Lucida Sans Unicode, 1993.

In 1997 Monotype was tasked by Microsoft to extend Arial to cover all of Unicode 2.0 — a mere 50,000 letterforms weighing in at 22Mb for the single font file. While the TrueType font format could theoretically support all of these characters in a single font file, it was a tricky process to make it work. We had to build many small 'fontlets' and then stitch them all together at the end of the process. My colleague of many years, Kamal Mansour, saw to it that the design was, as much as could be expected at that time, harmonious with Arial.

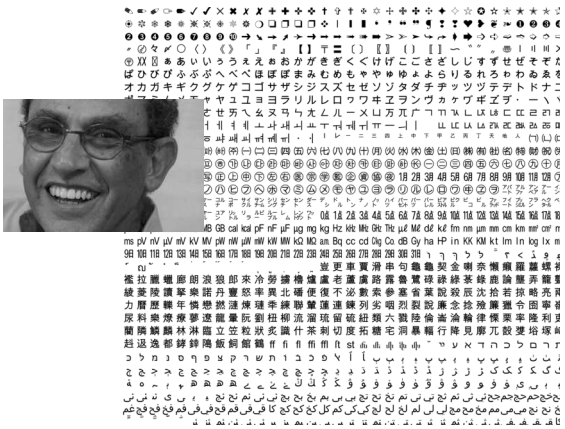

Figure 9: Monotype Arial for Unicode 2.0, 1997.

We included many more Chinese ideographs than were defined by Unicode. The extra 'glyphs' were included to support both simplified and traditional Chinese, so the actual number of letterforms in the font exceeded that of the Unicode standard.

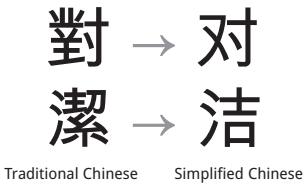

Figure 10: Same Unicode, different shapes.

The illustration above shows how the same Unicode character can be represented by two different glyph shapes. A Western equivalent might be to show the letter g in both a single loop and double loop form — it's the same Unicode character represented by two different glyph shapes.

In 2006 Google approached me to create a typeface family for a new mobile phone platform. Google wanted a unique UI experience for branding Android. A unique interface experience starts, of course, with the kind of typeface you interact with.

Because of Google's and Android's somewhat quirky branding I needed to draw an approachable typeface that was 'left of neutral'. Being too neutral wouldn't stand out as being unique to the brand. But if it was too cute or techno-looking the legibility and functionality would suffer.

Figure 11: New typeface design goal for Android.

Also, Android has a very specific rendering environment. Recall that cell phone screens were nowhere near the resolution of today's devices.

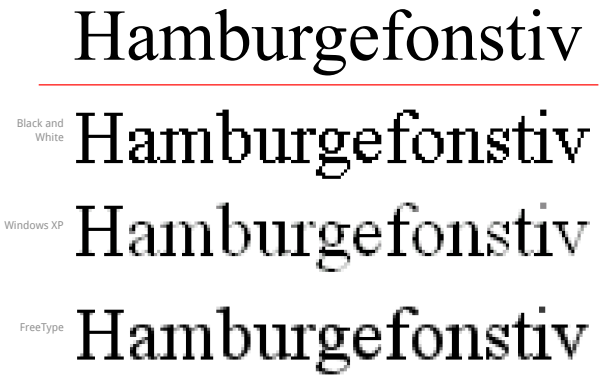

Figure 12: Different screens, different results, 2006.

This next example shows how the same design can be very different looking depending on the screen it's being viewed on. The thin parts of letters can look like they are disappearing or slightly too heavy, depending on how the software interprets and draws the letters.

These are some early drawings and experiments which were put into testing.

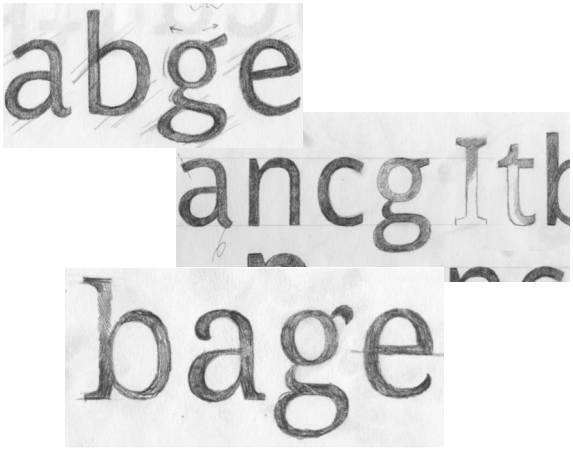

Figure 13: Early drawings for the Android font.

We had to create fonts and install them into devices to view the effects of small changes in design and proportion of individual letters. This was a laborious process because the user interface was being designed at the same time as the typeface was being developed. To complicate things further, the hardware which would become the first Android phones was also in the process of being designed and manufactured.

I designed the fonts with an eye on how they would render in various Android screens. I worked back and forth with their UI team to make sure there was enough contrast between regular and bold weights to aid in establishing a hierarchy in the interface. I made sure that detailing in each design was working well at these limited resolutions.

Once we were on the right track with the design we did some weight tests to see how much contrast was necessary between the regular and the bold weights.

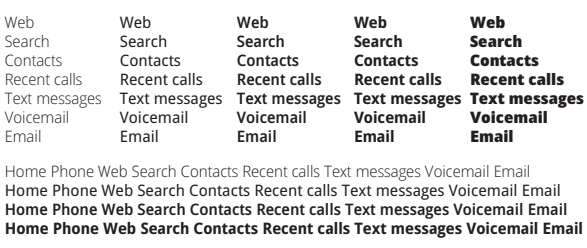

Figure 14: Weight tests for the Android font.

Only two weights were required; italics would be synthesized by Android in order to save storage space. On the other hand, after much discussion it was determined that a serif typeface should be part of the basic set of fonts. The serif fonts would be for reading news feeds and extended text. The sans would be for UI elements and menus.

Figure 15: Original Droid font family.

All of the fonts supported the WGL-4 character set but there was a sense that this was going to be expanded on if Android was successful.

The goal of course was to create a family of fonts which held up at small screen sizes and gave the platform an approachable, friendly appearance. We may have actually achieved this as a writer for Wired's online edition called the fonts 'googly'.

Figure 16: Droid display test.

When the time came to expand on what could be displayed in the Android UI, Google thought it was clear they did not want it to look like the left side below, regrettably similar to the insurance benefits statement shown earlier (fig. 4). Rather, they wanted a harmonized 'Android brand' look and feel across all the languages they were supporting. All of the scripts should have a contemporary, approachable and ergonomic feeling, closer to the right side. (Apologies for the typo in the Arabic text.)

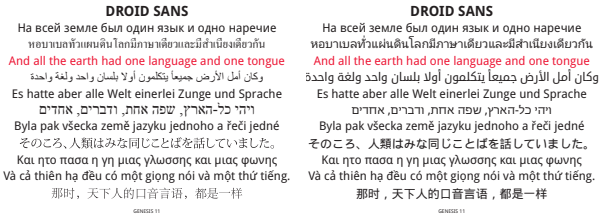

Figure 17: Unharmonized scripts on left; harmonized on right.

Making harmonized designs for scripts which have no historical relationship to each other is a bit tricky and in some cases nothing but a compromise of making things roughly the same weight. In Arabic, for example, the weight is on the horizontals rather than the vertical stems like Latin. This alone makes an enormous difference in balancing the weights. Visual cues can be picked up from the Latin shapes — the soft terminals and weight of the thin protrusions and the general contrast of thick to thin can be harmonized.

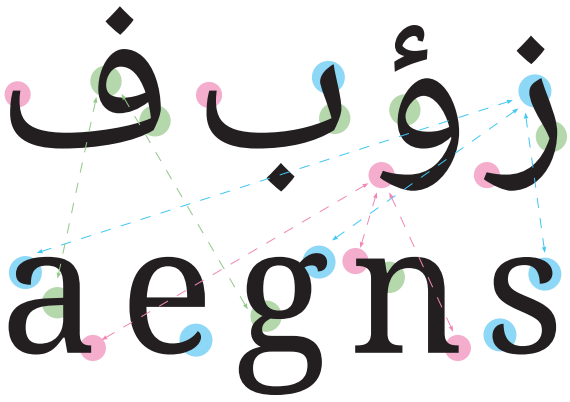

Figure 18: Harmonizations between Arabic and Latin.

The Arabic fonts were designed by Pascal Zoghbi with some art direction from me. The style that best matches Latin serif types is called Naskh. In most Naskh typefaces the counters are tiny, but in this case we exaggerated their size to mimic the Latin type's openness and aid legibility on screen.

In the example below, the top line of Arabic is in a style called Kufi which complements a Latin sans serif more closely. Typically, however, Arabic readers prefer the Naskh style (bottom line) for extended reading. I feel that it's similar to the resistance Western readers used to have for reading books set in sans serif typefaces. It really wasn't until the 1950s and 1960s that people started accepting this new typographic approach.

Working on this project I learned that Arabic readers were very accustomed to having to pinch-zoom text as soon as a page loaded. One of the goals

of exaggerating the proportions was to help prevent the need to zoom in to read default Arabic sizes. My understanding from Google was we achieved this in Droid and Noto.

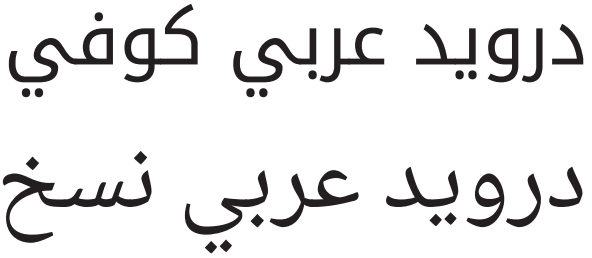

Figure 19: Kufi style (top), Naskh style (bottom).

Beyond Arabic, a Thai design was another early need for Android. This is serif style and it can be noted where some of the details are hinting toward the Latin serif typeface. (This and following examples are truncated on the left and/or right so details can be better seen.)

Figure 20: Droid Thai, with Latin for comparison.

A sample of the Droid Serif Hebrew — another script where the challenge is in the weight distribution being opposite that of the Latin.

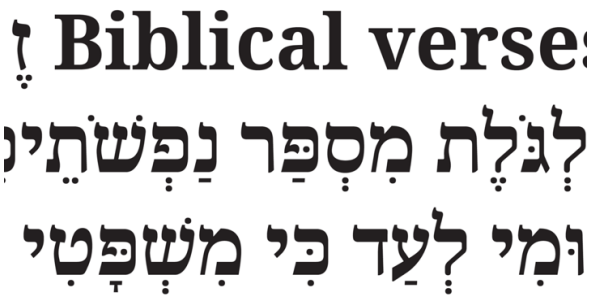

Figure 21: Droid Hebrew.

The Ethiopic script is often seen in a slanted form similar to an italic. I decided that an upright form would be most legible and useful for Android's user interface.

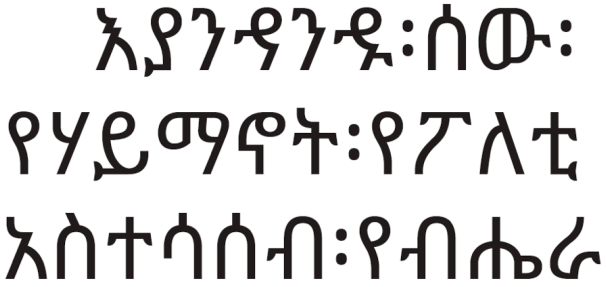

Figure 22: Droid Ethiopic.

Armenian takes many cues from the Latin lowercase shapes making it considerably easier to harmonize.

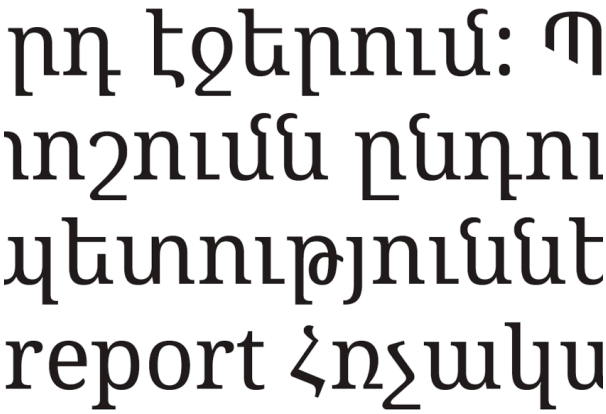

Figure 23: Droid Armenian.

Similarly, Georgian takes many cues from the round shapes found in the Latin. An entirely different texture than Armenian, but clearly a member of the Droid typeface family.

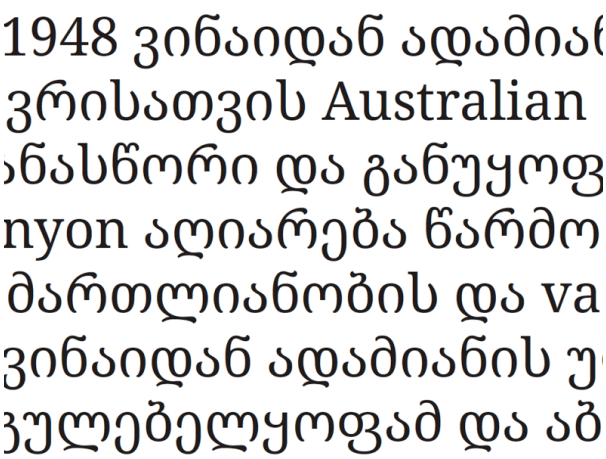

Figure 24: Droid Georgian.

Some may be wondering why I'm talking about Android at this point on the road to Noto.

Figure 25: Succession of designs.

In 2009 I was asked to adapt Droid Sans into a branding typeface for Chrome. This design became Open Sans, a slightly wider proportioned version of Droid more suitable for regular text in documents.

Roughly in parallel to this, the Chrome and Android groups at Google were discussing the idea of combining efforts on the development of a super font which would cover all of Unicode. The problem of 'tofu' displaying in Internet searches was becoming more problematic as the world's Internet usage was climbing dramatically.

With the joint packing of Chrome and Android, Open Sans then became the basis for Noto Sans and Droid Serif became Noto Serif.

By 2011 it was decided to expand on the Latin family for Noto so that it would include a full typographic palette of styles of weight and width. The sans and serif would have condensed and narrow versions added, additional weights from thin to black. The serif had an added range of contrast from super high contrast to low contrast. That's about 72 font styles per family — no longer would the polyglot typographic palette be limited by just regular and bold styles!

Figure 26: Noto, ca. 2011.

Unicode had by this time accounted for around 3400 characters to support Latin, Cyrillic, Greek and phonetic writing. This complement of characters supports over 500 languages. With these additional characters being added in all the additional weights, widths and style of Latin fonts meant drawing about 230,000 characters.

By now it was beyond clear that Noto would never ship as a single font binary with all of Unicode. Remember the old Arial Unicode font alone was 22Mb in size with just 55,000 characters. Instead, the Noto fonts are built as individual modules covering one or a few related Unicode script, or writing system, ranges.

Aside from the file size problem, writing systems beyond the Thai we drew become quite complex in form. Vertically, Thai is just barely able to squeeze into the vertical metrics of a Latin typeface. Below, you can see that Javanese (on the left) and Khmer (on the right) are even more complex vertically, and simply could not reasonably squeeze into the dimensions of the Latin script.

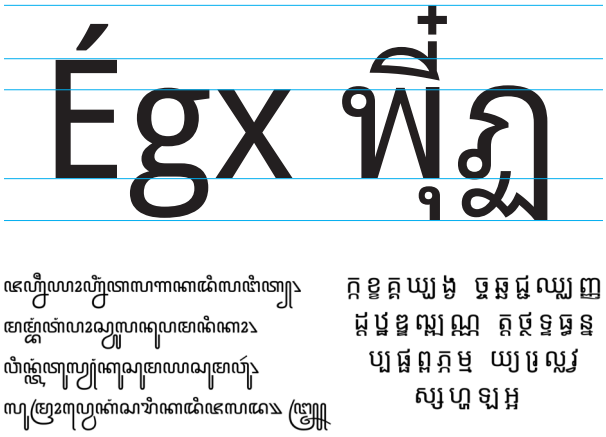

Figure 27: Beyond Latin metrics: Javanese (left), Khmer (right).

Another example is the Nastaliq style of Arabic used for Urdu, Pashto and Persian languages. While the commonly used Naskh follows a flat baseline, Nastaliq's baseline slopes downward to the left. The longer the word, the taller the dimensions become.

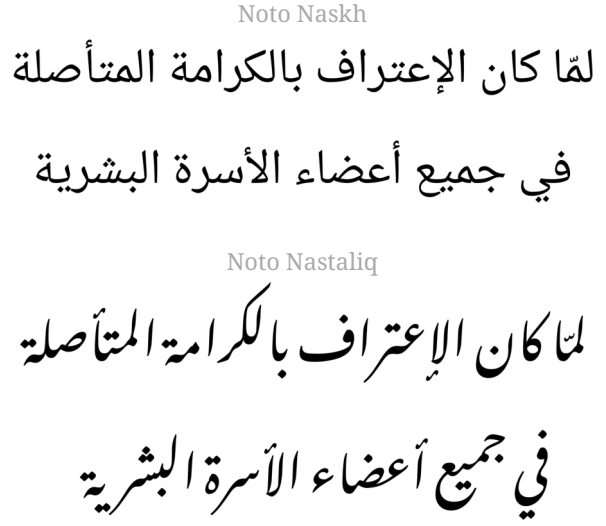

Figure 28: Noto Naskh (flat baseline) and Noto Nastaliq (sloped baseline).

One of the ways we looked at this enormous project was to break down the writing systems into categories. This way we could classify related scripts, either by their region, complexity, or relative use in the modern world.

Here, the orange boxes denote scripts that may have been dead for hundreds of years or more, while the greens are scripts used in India, and so on. This aided in prioritizing and organizing the expertise needed to complete each piece of the project.

Figure 29: Organizing scripts for Noto.

Cuneiform is a good example of the 'dead scripts' just mentioned; it may be as old as 5,000 years. While not in practical use, it is certainly useful for scholars and linguists to have encoded in a font file.

Figure 30: Cuneiform, original and Noto.

Anatolian hieroglyphs are at least 4,000 years old, thus predating Egyptian hieroglyphs. They are therefore represented as slightly more crude and smoothed over.

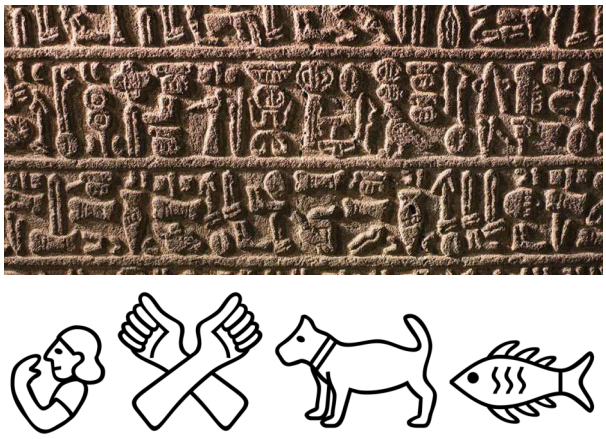

Figure 31: Anatolian hieroglyphs, original and Noto.

Egyptian hieroglyphs are much more crisp and refined in design than their Anatolian ancestors.

Figure 32: Egyptian hieroglyphs, original and Noto.

Many of the scripts are categorized as 'complex' scripts, requiring a great deal of programming to assemble words in the proper manner. Arabic, being right to left and having many forms of the same letter, falls into this class, as do the scripts used in India. Jelle Bosma is Monotype's creative lead in creating the Indic scripts, and is working on updates for Unicode version 13.

In the illustration below, the word on the right is the word 'Hindi' spelled out in Devanagari script. The top line is how it looks with plain Unicode characters set together, spelling out the word. On each successive line, you can see how the script is 're-shaped' as advanced typography tables rearrange the letters *as they are typed*. In the second line, the green characters change places. In the third line, the green and pink characters form a ligature.

Figure 33: Designer Jelle Bosma (left); the right shows the word 'Hindi' being shaped as it is typed.

The Indic writing systems appear quite different from each other but we've designed them to harmonize as much as possible. The rectangular Devanagari contrasts quite a lot with the fluid Sinhala, but their color and proportion are preserved to keep them in sync. It's the same with Telugu and Tamil; they contrast a great deal in overall texture, but their proportions and color tie them together.

Devanagari

अनुच्छेद 1 – सभी मनुष्यों को गौरव और अधिकारों के विषय में जन्मजात स्वतन्त्रता और समानता प्राप्त हैं। उन्हें बुद्धि और अन्तरात्मा की देन प्राप्त है और परस्पर उन्हें भाईचारे के भाव से बर्ताव करना चाहिए।

Sinhala

ක්‌රි.පූ. 500දී පමණ ශ්‌රී ලංකාවට හැඳින්වුණු පැරණි බ්‌රාහ්මීය අකුරු පැවත එන බව පුරාවිද්‌යාඥයෝ පවසන නමුත් හෙළ හවුල ප්‌රධාන පිරිසක් එම මතය බැහැකරයි. සිංහල හෝඩියේ අකුරු 60ක් ඇති අතර ඉන් 4ක් මෑත කාලයේ දී ඇතුළත් වූ ඒවා වෙයි. සිංහල භාෂාවෙහි ස්‌වර විශාල ප්‌රමාණයක් ඇත.

Telugu

ఆంధ్ర ప్రదేశ్ మరియు తెలంగాణ రాష్ట్రాల అధికార భాష తెలుగు. భారత దేశం లో తెలుగు మాతృభాషగా మాట్లాడే 8.7 కోట్ల (2001) జనాభాతో ప్రాంతీయ భాషలలో మొదటి స్థానంలో ఉంది. ప్రపంచంలోని ప్రజలు అత్యధికముగా మాట్లాడే భాషలలో పదమూడవ స్థానములోనూ, భారత దేశములో హిందీ, బెంగాలీ తర్వాత మూడవ స్థానములోను నిలుస్తుంది.

Tamil

தமிழர் உலகம் முழுவதும் பரவி வாழ்ந்தாலும் ஒரு சில இடங்களில் தமிழர் செறிந்து வாழ்கின்றனர். அத்தகைய இடங்களாகப் பின்வருவன உள்ளன.

Figure 34: Indic scripts in Noto.

Just as I mentioned there were technical issues to resolve way back with Arial Unicode; the same has been true with Noto. My colleague Toshi Omagari is shown below orienting his laptop to design Mongolian, a connected script which reads top-down, left-to-right.

The tool developers for GlyphsApp have been incredibly supportive in updating their product to make these complex scripts in Noto possible. It wasn't long before they delivered a fix which allowed Toshi to see his work in a way it would be used.

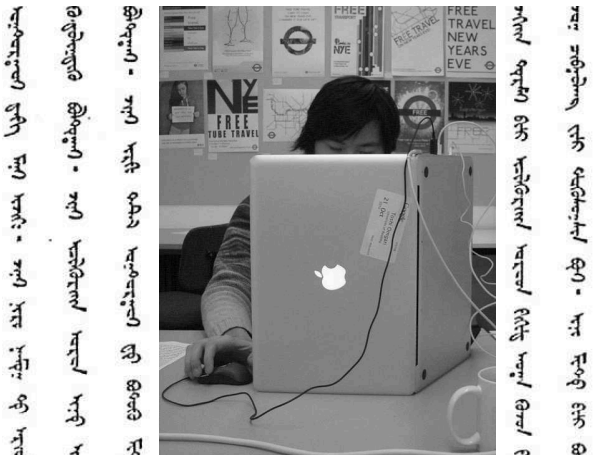

Figure 35: Toshi Omagari working on Mongolian.

Earlier I mentioned dead scripts. On the flip side is Adlam, a script developed in the late 1980s by the brothers Ibrahima and Abdoulaye Barry. This writing system transcribes the Fulani language spoken in Guinea, Nigeria and Liberia. Before Adlam, Fulani was written in either Arabic or Latin script.

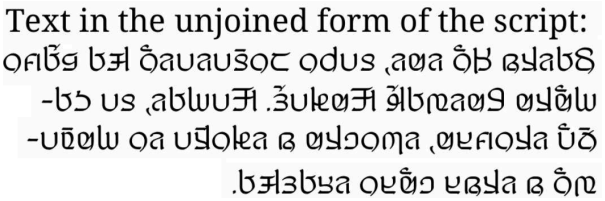

Figure 36: Adlam, developed in the 1980s.

By 2018 the Noto fonts covered nearly 64,000 characters and in the last 2 years there have been many updates. Unicode 13.0 adds 4 new scripts and 5,000 new characters to this count. "What could possibly be left?" you might ask.

Khitan, a language once spoken in Manchuria has been added, as has Chorasmian, a language of ancient Persia.

Sutton Signwriting, a notation system used to teach sign language has been added. It requires thousands of icons necessary to show hand gestures and facial expressions used by sign language interpreters.

And the Noto symbol font is getting many new characters including the long-awaited accordion and fondue dish symbols. The list keeps growing.

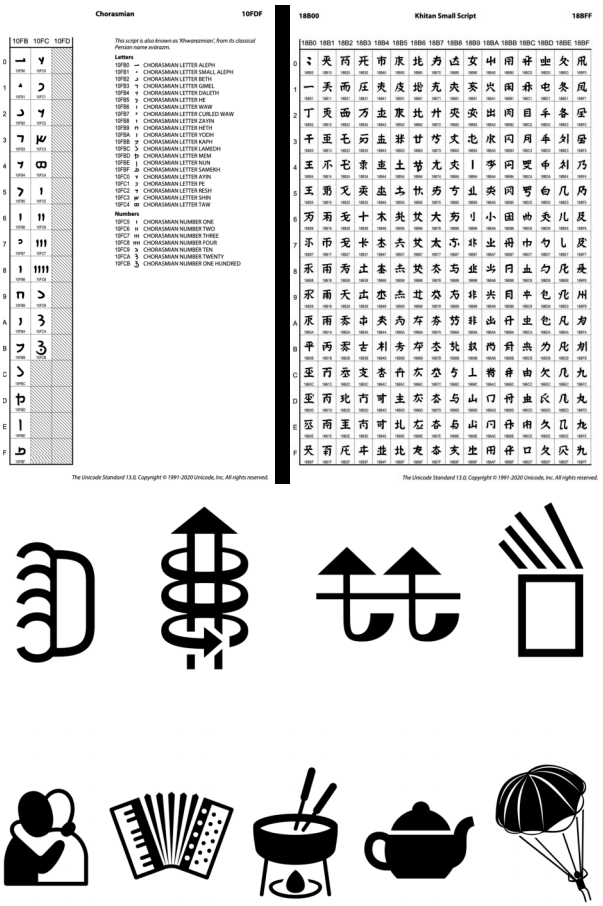

Figure 37: Added in Unicode 13.0: Khitan and Chorasmian scripts (top), Sutton Signwriting (middle), assorted symbols (bottom).

Noto will continue to be polished and refined to reflect the demands of the community of people using the fonts.

In addition to about 30 people within Monotype that have worked on the Noto project, we've been working with more than 30 outside linguists, consultants and designers around the world, notably including Fiona Ross, Tiro Typeworks, and Kigali Design.

Other people I'd like to thank here: Abdoulaye & Ibrahima Barry; Jo De Baerdemaeker; Cadson Demak Ltd; Diane Collier; Fontef Type Foundry; Kalapi Gajjar-Bordawekar; Yanone Gerner; Gajjar & Vilhjamsson Private Limited; Kimya Gandhi; Patrick Giasson; John Hudson; Indian Type Foundry; Yanek Iontef; Letterjuice Ltd.; Ben Mitchell; James Montalbano; Elena Papassissa; Rainer Erich Scheichelbauer; Zachary Scheuren; Georg Seifert; Vaibhav Singh; Anuthin Wongsunkakon; Pascal Zoghbi.

The community of users and testers who have provided feedback is, of course, much larger. It is an honor to be working for them to make this enormous undertaking a possibility.

In the end Noto may not be used for retranslating the bible or imperial decrees. It might be very simple messages that we can convey with this enormous tool we have at our disposal. And maybe Noto is another step towards that romantic notion I heard as a student — that people around the world could spend more time communicating instead of fighting.

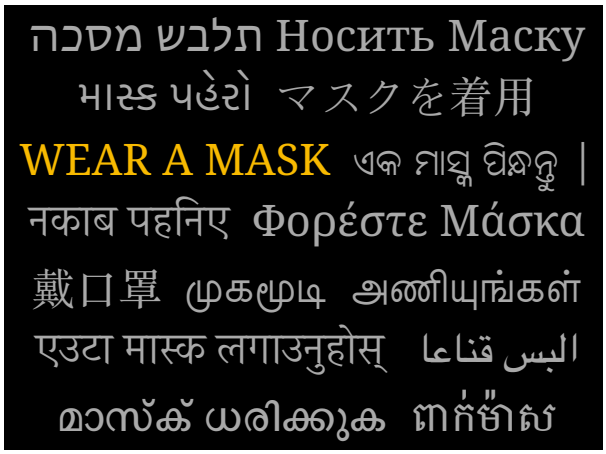

Figure 38: Translations unedited, via Google Translate.

◇ Steven Matteson
Monotype
monotype.com/studio/steve-matteson