1. INTRODUCTION. An ongoing goal of language research should be to ensure that the research creates the opportunity for the target language to return to vital use in the community. In contemporary society, the availability of fonts for community use can make the difference between language that sits in books on dusty shelves, and language that is used to write letters and essays, to create posters, notices, and newsletters, and be used in education, politics, and business. In early 2000 I began working with Athapascan languages, specifically Chiricahua, Mescalero, and Lipan Apache. At that time only one font was available for Athapascan languages, and the cost was $125. This was expensive compared to the large number of fonts distributed freely with computer operating systems, and was cost-prohibitive for people living on reservations. In addition, a single font available only from the Summer Institute of Linguistics (SIL) did not provide users the same kind of creative opportunities that are commonly available to users of dominant languages. Sorting and spell-checking are essential functions within language-related technology in contemporary communities. In addition, I wanted to be able to create interactive dictionaries for use on the web, and this required database support. Many special fonts are ornamental and do not support accurate sorting, and thus cannot be used in dictionary development because special characters are not recognized by word processing programs.

Because of my own personal experience and preference, and the nearly ubiquitous availability of the Microsoft® Office suite of programs, I did my initial font development work using this system. Microsoft Word provides scripting and template functionality, while Microsoft Excel supports both creating and reordering word lists. Microsoft Access supports a database in which dictionaries and related language information, including recordings, movies, and animations, can be prepared for use on the World Wide Web. The schools we were working with at the time had Microsoft Office available for both students and staff.

2. ATHAPASCAN LANGUAGE CHARACTERISTICS. Southern Athapascan languages—Navajo and Apache—use eighteen special characters that are not available in commonly distributed fonts. Five of the characters indicate rising tone—á, é, í, ó, ú—and can be constructed using a control key sequence. Ten additional vowels—ą ę, į, ǫ, ų (high tone, and high tone nasalized vowels, respectively)—are not easy to construct using a conventional keyboard. The remaining three characters are the glottal stop, typically represented using an apostrophe, the high tone n (ń), and the barred l (ł).

For dictionary development, vowels must sort in the correct order, which requires that they be recognized as unique characters rather than simply as specially accented vowels. The apostrophe (glottal stop) must also be recognized as a character rather than as a punctuation mark. In addition, for developing contemporary learning materials using software such as Adobe/Macromedia Flash, which allows us to create immersion-style language
materials, the fonts needed to represent Athapascan in programs other than Microsoft Office. In all cases we needed an easy way to input the characters.

3. ESTABLISHING DESIGN GOALS. When I began working on this challenge in early January of 2000, a review of available fonts did not produce a wide variety of choices. One proprietary font was available from SIL. Another font was being distributed on the web by the San Juan school district in Utah, but had the disadvantage of having replaced the internal locations for the numbers 0–9, and some of the special characters—dash, equal, left and right bracket, and backslash—to represent the special Athapascan characters. This made the font unsuitable for spell-check, for database applications, and also for any mathematical or scientific applications that depended heavily on numbers and numeric functions. Microsoft Access had the additional constraint of allowing only one font per table, which eliminated the option to use multiple fonts in the development of mathematical and scientific materials. Given this situation, my only available option was to create or modify fonts to satisfy the specific design criteria.

The challenge became one of developing a font collection that could be distributed to those who needed it, would be usable in a wide variety of programs, and would support contemporary functionalities such as sorting, spell-checking, and database update and retrieval. In addition, character input had to be simple and easily understandable by a wide range of people, many of whom would not be technology experts. Finally, we had to have fonts that would function in a wide variety of programs, that would work on both PC and Macintosh platforms, and that could be included in saved documents. In Microsoft Word, this meant that the fonts had to be True Type (In Microsoft Word, see Save As/Tools/Save Options for font embedding options).

4. PROPRIETARY VS. NONPROPRIETARY SOFTWARE. This article discusses software platforms that run on Windows and Macintosh operating systems. When we first began working, these were the two most popular desktop operating systems in use on reservations. Our goal was to create tools that could be used easily and required just basic expertise. UNIX/Linux operating systems were also popular, but had several disadvantages. First, they were released as distributions, and these required technical knowledge to install. Second, they were not in wide use on reservations. Finally, there were no popular font development tools available for the platforms.

There is a common conception that “open-source” software is perhaps the optimal choice, either because it is free, or because using it diverts funds from corporate coffers. While this is sometimes true, to my mind, the reliability, customer support, and ease of acquisition, (including the possibility of charitable and grant-funded donations and acquisitions) of strong, broad-base, commercial software far outweigh the potential benefits of software that is not widely used, and that does not have strong, reliable customer and technical support. Each of the limitations of open-source software comes with its own attendant difficulties, and we were not in a position to deal with platform challenges as well as tool development challenges.

Because we were a small shop, working with little-used languages, we needed to make the best use of the resources we had. There were limitations in working as a small shop: many things needed to be developed, several people needed to be trained in both the tools...
and the philosophy and theory, and, since we were working on reservations, lots of travel was necessary. Thus, having a developmental platform that could provide additional, comprehensive support and training was a necessity. So we opted for what seemed to us a natural choice. We developed strategies and tools that would help tribes use the resources they already had to support the development resources in their own languages.

An additional benefit of working with Microsoft Office, and Microsoft Word in particular, was that it supported a scripting language that we could use to support different functionalities for materials development. Accessing the characters that I created for Southern Athapascan languages was difficult, since nearly all of the two-character shortcut codes had been used by keyboard enthusiasts in the software configuration process. This relegated our user functions to three-character codes, and this was simply not acceptable for the people who would be using our tools. Microsoft’s scripting language let me create a template that would insert special characters when the icon for that character was clicked. The process was not as rapid as keyboarding, but it eliminated the challenges of memorizing and using multiple keys for characters.

5. UNICODE AND THE UNICODE MYTHOLOGY. An investigation of the intricacies of language support in Microsoft Word revealed software dependencies on both the Language Standard (ISO 639/639-2) and the Country Code Standard (ISO 3166/3166-2). At the time, the three-digit ISO 639-2 had not been released, and there was no way to specify Apache or Navajo language as a default option in the Microsoft Word language options list. Later, ISO standard 639-2, with its three-digit reference codes, was released. One can now identify Apache languages as ana, but not as Chiricahua, Mescalero, Lipan, or Kiowa, and this precludes automated grammar and spell-checking. With the new codes, Navajo is identifiable as nav. However, in the Unicode character set, there has been no accommodation made for the special characters required in the contemporary representation of Southern Athapascan languages, nor does the Unicode Consortium have any current intention of solving some of the challenges that these indigenous languages present.

In discussions with Unicode Consortium members, it was clear that they had convinced themselves that we can “merely” compose our characters by selecting the codes for each accent mark and composing them as we go. This is not sustainable because of (1) the sheer labor involved in such a compositional scheme when working with languages like Athapascan; (2) the complexity of learning and understanding the process for people whose main goal is not to master technological intricacies but to produce materials in their native language; and (3) the requirements for sorting: characters must sort according to a consistent evaluation scheme, and this is not possible if the definition is composed of two or three different Unicode characters, each represented by a component code. For those unfamiliar with the computer-based sorting process, characters would be sorted by the first specified component code, which could be the letter, for example a, or one of the diacritics. This character would then be sorted by the second component code, which again could be the letter or a diacritic, and then the third component code, if it existed. This type of character specification provides no reliable sort sequence. It also creates real difficulties for spell-check, dictionary development, and database development and retrieval.

Where the characters are located in the font table is also in question. The font table is actually divided into categories of characters, and these categories are often indexed by the
language and country codes. Although word processors allow a choice of languages, if the characters for indigenous languages occur in multiple and significantly different language sets, accessing them in normal text processing will be challenging at best, and impossible at worst. Also, the language standard for country code allows us to index down to the U.S., but not to the reservations, so we can’t set a default language code for Navajo Nation, or for the Mescalero or White Mountain reservations, because we don’t have unique country codes for them, and the default language code is included in the parameters for the country code.

So inside various word processors are a vast complex of tables, standards, and procedures that define and constrain how the functionality appears externally. It may not seem so at first, but it is really very complicated, which was another consideration in deciding to choose a more popular rather than less popular platform. Approaching problems on a broader scale, for example with word processing programs that were used world-wide, offered a greater potential for success with organizations such as the Unicode Consortium than would be possible with software that was used on a more limited scale.

6. INTRODUCING FONTOGRAPHER. Fontographer, developed by Macromedia, was one among only a few choices for font editing software available at the time. Other programs were intended for professional font designers and were both more complex and more expensive. I needed a program that would be easy to understand and use, yet would provide the functionality necessary to create Athapascan fonts. Fontographer was also one of the premier font editing programs and was highly regarded during a time of dynamic change, growth, and rapid development of font types. It also provided the possibility to integrate

![Figure 1: General preferences selection window](image-url)
Unicode functionality into the fonts themselves and subsequently into the programs that used them.

Fontographer provided all the necessary options and functionality. It would read both PC True Type and Macintosh font files, abstracting the font information into working files. It supported a wide range of user options and preferences. Figures 1 and 2 show the General and Point Display preferences windows. These two figures demonstrate the broad scope of decisions that must be made when dealing with fonts. Figures 3 and 4 show the internal details of modifying fonts and their “characters.”

![Point Display preferences window](image)

**Figure 2: Point Display preferences window**

Figure 1, the General preferences window, provides a choice for the common option of processing *undos*. It also shows the font-encoding choices that must be made when fonts are used and when they are opened in Fontographer. These preference choices show the strong influence that the operating system, in this case Windows, can have on how fonts are recognized, handled, displayed, and possibly recoded. When special fonts are being developed, it is important to understand, sometimes by trial and error, how the different options will affect the performance of the font.

Since I was creating special characters for Athapascan, I wanted those characters to be used as I designed them, so I always selected the options that used the font encoding rather than allowing Windows to select encoding or recode characters.
Figure 2 shows the many options available for glyph editing, and at the same time creates the opportunity for a brief discussion of the structure of fonts and the language of the font domain. Text characters are actually composed of two separate types of information: the numerical information that identifies them, and the graphical information that defines their appearance. Although we commonly refer to them simply as *fonts*, modern fonts are actually computerized representations of *typefaces*. Typefaces are designed, and hence have a style that constrains their appearance, in addition to characteristics such as weight and italicization. These characteristics can be applied internally, in the font-face design, or externally, as a numerical transformation of the font. Such choices, and their attendant complexity, increase in importance as one’s knowledge of fonts and their design and application increases.

For the new user, figure 2 implies that character glyphs—the figures that define how a character looks—are composed of lines attached to points. In most cases, and always for scalable fonts, this is true. The exceptions are the older, bit-mapped fonts that were not scalable. A characteristic of scalable fonts is that they can be mathematically recalculated to retain their stylistic beauty and visibility at different sizes and as bolded, italic, or bolded italic.

Getting used to the point-line methodology takes some time, and can be characterized as an art rather than as a science. In my initial experience with Fontographer, the goal was to develop fonts that would be used in the development of immersion learning materials, so I was not as concerned with copyright distribution matters as I would later become. Because my first goal was simply to create composite characters and determine the specifics of their application scope, I did not have to be as concerned about working with the actual points, lines, and hinting as I would later become.
Characters can be viewed either by character key or hexadecimal representation, as shown in figure 3. The hex representation is very useful when searching for locations to be modified. Fontographer can generate both True Type and Macintosh font files, and provides both easy and advanced interfaces for different levels of generation specification detail and control. Initially, these functions were the most critical, because the goal was to create new character combinations from existing fonts. The edit functions copy, copy widths, and paste were also essential.

Figure 3 shows the character layout in Fontographer. I usually work with the Unicode or hexadecimal display, since this makes it easier to find character locations in large font files.

One advantage that I had in learning to use the program was having a clear set of design goals and objectives. These goals and objectives became part of the model of expectations I had about what I should be able to do with the program, how the resultant fonts should work, and how I would be able to test them. Being guided by an expectations model of this kind makes learning much easier than approaches that identify the different program functions while leaving learners to develop a model of understanding on their own.

7. DEVELOPMENT EFFORTS. On the first try, I created the special characters in the designated user area. To do this, I first opened an existing font, copied the vowels and the accents that already existed, and combined them in the new locations. I then generated the fonts and installed them in the Fonts folder. This font configuration did not work in Microsoft Office because the characters were not recognized as characters used in text creation. Instead, they were considered ornaments, treated as punctuation, and ignored.

With Fontographer, I was able to make this discovery in under two hours. This included installing the software, choosing a font to modify, mastering the edit functionality, and generating an output font.

On the next try, I first examined existing fonts to locate characters that would be recognized as text characters in Microsoft Word, but which were not used in English or Spanish and so would not interfere with normal, day-to-day text creation and modification in the American Southwest. I decided to reuse some of these locations, and see if the resultant fonts would produce the results I needed. In fact, this approach did work. By replacing certain character glyphs with my newly constructed ones, I was able to create fonts that allowed me to use the spell-check function in Microsoft Word, and create and sort dictionaries and database files.

The next major development task was to provide a mechanism that would allow people to use the characters in documents easily. The usual way to input special information not immediately accessible via the keyboard is to use combinations of control characters, alphabetic keys, and other special characters. An examination of the control key combinations in Microsoft Office revealed an extensive collection of two-character control key sequences. This forced me to use three-character control key sequences to identify my special characters. Remembering and using three-character sequences did not fit the ease-of-use profile that was part of my design specifications for Athapascan fonts. In the same vein, control key sequences were not always supported by the graphics and materials development software that I would be using, so I looked for alternatives.
Microsoft Word provided both scripting and a menu template option. Using these tools, I could design a template that would show the character itself as an icon, and that would contain the script to insert the character into the document when the icon was clicked with a mouse. This choice had the advantage of being easy to use in terms of inputting the characters, but had the disadvantage of being slower than typing using fonts that replaced the numerical keys, as described above. However, the template plus the newly-designed Athapaskan fonts still satisfied the bulk of the functional specifications. For my goals, characters that would support spell-check, sorting, and database creation were preferable to characters that could be input quickly but would have little functional capability beyond their immediate visibility. Besides, technology offered other solutions for this particular challenge. I decided to use the template, which hid the three-character control sequences from view and provided a graphical user interface that was intuitive and easy to use.

Once I had accomplished all of the initial design goals, I realized that I had definitely created fonts that could provide extensive creative options for Athapaskan language documents, had multifunction capabilities, and were easy to use. But what I also had was the European aesthetic: Modern fonts are based on font designs that have evolved over hundreds of years in England, Germany, France, and more recently, the U.S. and other parts of the world. In other words, I had merely pasted the Athapaskan functionality onto the European aesthetic. In addition, fair-use guidelines allow font owners to modify fonts for their own use, but not to distribute them. Although the technology was fairly easy for me to use, it would not be easy for tribes to use the same approach to creating fonts for their use. I needed another solution, and creating fonts of my own, or modifying fonts that were not copyrighted offered these opportunities. In order to accomplish this goal, I needed to learn more about how to modify fonts and glyphs.

Font information can be changed in two ways. The overall font characteristics can be changed by using what Fontographer characterizes as Font Info. This includes the style (bold, italic, etc.); the M-square units, which is defined as “a square the size of the capital M”; the ascent and descent, which are a font’s maximum allowable distance above and below the baseline respectively; the different parameters for True Type and Postscript fonts; and copyright and notice information.

The second way in which fonts can be changed involves changing the actual shape of particular glyphs, which are the physical face of the character. When these glyphs are changed, the designer must make the decision about whether all characters of a similar type—for example, whether the rounded parts of a, e, g, o, and q should be the same or similar, or whether only one character will be distinctive. These are stylistic design choices. Traditionally, font designers have striven for what is called a classic style of beauty, where each typeface has an overall stylistic theme, and each character fits within that theme. Popular typefaces that exhibit these characteristics are Arial, Times and its various derivations such as Times New Roman, and Goudy Old Style. However, with the availability of font programs such as Fontographer, new designers are making different choices. The aesthetic sense is changing from the more sedate traditions to more exciting typefaces such as Comic Sans and “handwritten” fonts derived from individuals’ handwriting. I own a font that has been designed using petroglyphs, which is featured on my web site; a font called Antlers, where the designer embellished individual characters with antler designs; and, a font called Critter, which creates the character glyphs using images of animals. In
contemporary font design, we are emerging as designers of fonts for contemporary uses, meanings, and applications. Where we are lacking in this landscape of creative fonts is in collections that represent and reflect the American Indian and world indigenous peoples’ sensibilities, as font development by indigenous peoples has yet to be undertaken on any noticeable scale.

Another consideration when making the decision to modify or redesign fonts relates to intellectual property and fair use. The fair-use guidelines allow font owners to modify fonts for their own use, but not to distribute them to others. Agreements must be negotiated with the font factories when existing fonts, such as Arial, Times New Roman, Lucida, or Georgia, are to be modified to support indigenous languages. Fontographer provides nine royalty-free fonts. Additionally, royalty-free fonts are often available on the web. When choosing these fonts for adaptation, you must read the notices carefully, because some allow you to adapt the fonts for your own use, but not to redistribute. Others allow redistribution so long as their intellectual property information is included. Fontographer provides access to this information, as well as to the physical characteristics described above, in the Font Info windows.

I needed to be able to distribute the fonts, so, having mastered the basic Fontographer functions of minor editing and font generation, I decided to try my hand at design. I chose a royalty-free font, and began to modify its various properties, generating the fonts and examining their appearance in “template documents” that I had created for the purpose. These template documents contained the complete collection of special Athapascan characters. To use them, I would simply change the representational font using the font selection box in Microsoft Word, or install the new font and see how the letters looked. I examined the different font sizes (since these are devised mathematically), the different

![Image](image-url)

**Figure 4:** The Edit window in Fontographer. Selected character is high tone, nasalized e
weights, and the different font formatings such as large and small capitals, and sub- and superscripts. This allowed me to see how well my changes were working in the overall visual domain.

The results of these efforts were Athapascan Bitsine and Athapascan Naaki. Figure 4 shows a character composite in the Athapascan Naaki font that I designed. The special design of the e glyph is characteristic of this font, and is suggestive of American Indian graphics and petroglyphs that represent mountains and high places.

The glyph edit box is opened by double clicking on the character in the font display window shown in Figure 3. In Figure 4, the cedilla is selected; each of the + signs indicates a location where the graphic can be grabbed and resized or reshaped. This configuration resizes the overall size and shape of the selected graphic, while shape is changed by moving individual points (not shown).

When one designs fonts, the number of options, possibilities, aesthetics, and approaches increases vastly over the simple approach to constructing special characters for underrepresented languages. Fontographer supports all of these, while maintaining its easy-to-use approach.

8. SUMMARY. I like this program very much. Part of this is because Fontographer accomplishes all the tasks I set for it without requiring a convoluted or difficult learning path. My appreciation also derives from its familiarity, since I have used it continuously for nearly seven years, and have developed fonts for other languages as well as the Southern Athapascan family.

Pros: The program is old enough that it doesn’t install in the registry. It is comprehensive, and a good balance between robust functionality and ease of use. It responds quickly, and generates font files rapidly for both Macintosh and PC.

Cons: There first update in nearly 10 years was on March 31, 2006. The upgrade to Version 7.7.3 is free for 4.7.x users, and $99 for Version 4.1 users.

The free fonts are clunky and need a full redesign to be serviceable for a broad range of media.

Primary function: Font creation and modification.

Platforms: Windows and Macintosh.

Source: Fontographer is a commercially available product originally published by Macromedia and now available from Fontlab. The price is $349; academic and volume pricing is available from Fontlab (www.fontlab.com) when you call and speak to a sales representative.
**Reviewed version**: 4.1, released March 1, 1996.

**Application size**: 1,794 KB; associated files are 1255 KB; nine royalty-free font files consume another 46 KB; The help file is 245 KB.

**Documentation**: Documentation is comprehensive and provides information both about using Fontographer and also about theoretical aspects of font development. The documentation is brief, but comprehensive and to the point. It is an old style .hlp file using simple hyperlinks and does not have the more sophisticated interactive skinning of more recent productions. However, it does provide all the information one needs to use the program and to identify topics for additional investigation that would facilitate development of specific user projects.

The documentation also includes a glossary of terms that are hot-linked appropriately to the documentation.

**Author’s Note**: The two fonts mentioned in this article, Athapascan Bitsine and Athapascan Naaki, as well as the template and installation instructions, are available for download on the author’s site at http://learningforpeople.us

Mia Kalish
MiaKalish@learningforpeople.us