https://doi.org/10.20378/irb-49404

Complex Script-Systems on Today's Personal Computers

Sebastian Kempgen

0. Introduction

The present paper is based on the experience gained by the author over the last ten years from designing fonts which today are used by many Slavists, especially in Western Europe. These fonts include "Method" (Cyrillic Old Bulgarian/Old Church Slavonic), "Kirill" (Bulgarian round Glagolica), and "Glagol" (Croatian square Glagolica) along with Cyrillic typefaces, transliterations fonts etc.

1. OCS – The Situation

Old Church Slavonic (OCS) or Old (Mediæval) Bulgarian texts are characterized, among other things, by the following features:

- No normative alphabet or orthography;
- No clear-cut distinction between grapheme and allographs;
- Many diacritics above lowercase and uppercase letters;
- Complete alphabet used as superscripts;
- Many ligatures.

Examples illustrating these points are well-known to every specialist. In fact, each scribe or school may be considered to use his/its own system applying more or less strict writing rules to the text in question. The farther we abstract from individual manuscripts, the closer we approach the ideal OCS alphabet which text-books and grammars present, although hardly any single text can be said to implement it one hundred percent.

Consequently, a font-designer has to make a decision about what he is aiming for: is it an individual text which he wants to electronically represent or is it the normalized abstract orthography and appearance of an OCS grammar? If a grapheme has several letterforms, are both to be included in the font or not? Which one has to be chosen as the basic one? Is it the older form, for example, or is it the more common variant? How should diacritics and superscripts be realized?

2. Font Technology

The basic distinction here is between bitmapped fonts and outline fonts. Ten years ago, bitmap fonts were the standard, when dot-matrix printers were the most common printers and laser-printers did not yet exist. In bitmap fonts, each

character is represented at a fixed point-size by a collection of pixels. Today, of course, outline (or vector) fonts are the state of the art. The difference between bitmap fonts and outline fonts is comparable to using a painting program (which turns single dots on or off) or a drawing program (where one draws objects which can be re-sized at will). In outline fonts, the designer draws the outline of a character, setting corner and curve points and connecting them with lines. While this sounds simple for a single character, it becomes a very demanding task for a complete alphabet. The premier font-creating package is called Fontographer; other programs (such as FontStudio) have practically disappeared or occupy a niche only (such as URW's fine Ikarus program). Using such a font-editor, the designer generates the end-user product, for example a PostScript Type 1 font or a TrueType font. When these fonts are being used, the outlines contained in the font files are scaled to the appropriate size (say 12 points) and filled with pixels, depending on the resolution of the output device. Laser-printers, for example, have at least four times the resolution of standard computer monitors. Consequently, characters are printed with much finer detail than represented on-screen. Bitmaps may or may not be required with fonts, depending on the platform and the font format. Commercial fonts usually come with a range of hand-edited or polished bitmaps for perfect on-screen representation, especially in small point sizes.

Font files as used by Fontographer usually contain 2^8 = 256 slots of which 225 positions may normally be used. Although font files with 512 or 1024 characters are possible, creating fonts with that many characters is impossible if we deal with single-byte languages. Thus, in all practice, we are limited to standard 8-bit fonts. If the number of characters required exceedes this limit, as is easily the case with OCS, several fonts have to be created with each one being dedicated to a specific task. For example, one font can contain the basic alphabet and diacritics, another one superscripts and so on.

3. OCS Fonts on Today's PCs

3.1. Encoding Strategies

Font development is always done under specific historic conditions. The basic and simplified decision algorithm when creating a font for a specific language or script can be summarized as follows: Is there a standard? No – Create new encoding. Yes – Is the standard practical? Yes – Follow standard. No – Create new encoding.

For OCS, the answers will be "no" or "yes"/"no", in both cases resulting in the advice to develop a new encoding. It is reasonable to base an OCS encoding on the encoding chosen for Scientific Cyrillic, and this in turn is based on ASCII or its variants. After an encoding for OCS has been decided on, Glagolitic can also be encoded: ASCII → Scientific Cyrillic → OCS → Glagolitic. The author himself, for example, created his own encoding scheme for scientific Cyril-

lic in 1985; from this in 1988 the encoding for Method, the OCS font, has been derived. And on the encoding used by Method, the corresponding Glagolitic fonts, created in 1992, are based. This ensures – as far as possible – compatibility between the different scripts.

When choosing an encoding for new characters or scripts, several principles can be followed when starting from and modifying the standard Latin 8-bit-encoding:

• Vowel on vowel, consonant on consonant

The reason for this is two-fold: it is a mnemotechnic aid to remember where the characters have been placed on the keyboard, and it ensures maximum compatibility with existing hyphenation routines.

• Character pairs on character pairs

This principle is also obvious, but it is not always followed when it could be followed, and sometimes it is impossible or impractical to follow. Again, there are two main benefits: keystroke pairs remain pairs (lowercase vs. uppercase), and functions such as "small caps", which most wordprocessors offer, will continue to work.

• Basic ABC into lower ASCII

The reason for this is to make access to the most frequently used characters as convenient as possible, because the lower ASCII character set can be completely accessed on nearly every keyboard. Variants and accented characters can be put into higher ASCII. That less often used characters are somewhat harder to type than the basic set is a principle which underlies many keyboard drivers and thus is nothing specific. (When designing keyboard drivers, this principle becomes "Put basic ABC on normal layers, additional characters on option-keys.)

• Phonetic or graphic similarity

When Cyrillic is mapped to a keyboard layout, phonetic similarity may be used. Thus, for example, Cyrillic А will be on A, Б on B, Д on D and so on. This will, of course, depend on the phonetic values usually associated with characters. On the German keyboard, for example, you have characters *ä, ö,* and *ü*. Putting Cyrillic *я, ё,* and *ю* on these keys is an obvious solution for German users.

• Design for local keyboard driver

This principal is especially popular with Macintosh users, because the Macintosh has always had very convenient keyboard drivers. On the Macintosh, the Option (or Alt in PC terms) key by itself is not used to issue commands but is used to enter foreign characters, mathematical symbols etc. in a very straightforward manner. For example, on most keyboard drivers, option-a will produce *å*, option-c produces ç, option-p produces π, option-z produces Ω (which, of course, is the last character of the Greek alphabet) and so on. Macintosh keyboards thus have four layers: lowercase, shift, option, shift-option. As a keyboard has around 50 keys, this allows access to 200 characters. Add the five

accent keys which work as dead keys to enter accented characters, and the complete 8-bit character set can be can accessed very easily. Typing ALT-0 + decimal value has never been necessary on the Macintosh.

This feature of Macintosh keyboard drivers makes it clear why designing fonts for local keyboard drivers is so popular on this platform. Option keys produce alternate or related characters, and if a foreign language has more characters than the basic English alphabet, it is natural for users to access less often used characters by using the option key.

There is a drawback to this solution, however: what may seem logical on one keyboard, may not be true for another local keyboard driver. While many option-keys are identical on all international keyboads, most shift-option keys are not. When designing for the US keyboard, a certain combination may not make much sense on the French keyboard, for example. But as keyboard drivers can be easily created, copied, modified and installed by end-users on the Macintosh, a simple solution is to include the keyboard driver for which a font was designed with the font itself.

3.2. Character Design

Character designs have a "look and feel", and this is also true for OCS. Different countries have different traditions with respect to the printing types they use, so what is "normal" for a user in one country will not necessarily be of the same quality to another user. Font designers usually call OCS letters a "black letter design" because of the thick black stems which mediæval scripts exhibit giving OCS fonts a certain "weight".

When designing a general-purpose OCS font, it is reasonable to start from the printing types as used in standard grammars, text-books etc. because the look of these characters is what most normal users will expect from an electronic rendition.

However, specialists might also want to reproduce the look and feel of a specific text they are working on. Consequently, fonts can "look" Russian, Serbian, or Bulgarian, and/or 12th, 15th, 17th century, for example. Taken to the extreme, one font could be created for every single text.

After the basic decisions have been made, the process of recreating characters electronically sounds simple: select the letter form for each character which seems most typical, use a scanner to bring a picture of the character into the computer, then use a font-editor to trace the outer edges of the character. While this can be done quickly for a single character, expertise, skill and a trained eye are required for complete character sets of commercial quality. High-quality fonts must, for example, exhibit even character size, spacing, constant "colour" etc. when printed. The amount of work that goes into designing a new font from scratch has been compared to writing a book. Also, hardware requirements should not be forgotten: font design requires the use of the largest

possible monitor (anything else but a 21″ flat screen is too small), a fast CPU with a large memory (RAM), and a fast PostScript-printer.

3.3. Diacritics

In standard encoding schemes for Western languages and now for Eastern Europe, all foreign characters are preaccented, that is, are available as such. Coupled with corresponding keyboard drivers, this, of course, is very convenient to users, and it is also the only way to achieve a typographically precise placement of the diacritic in every instance. It is important to remember that by pressing first ` and then i on the keyboard, a user does not tell the computer to put the grave accent on the i, rather, he tells it to call up the preaccented ì (notice the absence of a dot over the i!).

In contrast to standard modern languages, OCS has a very large number of different diacritis which can be combined with most base characters. To construct every conceivable combination of base character plus diacritic as such would result in a very large character set which could hardly be handled any more. Literally thousands of combinations have to be taken into account, because diacritics may sit centered over a character, centered between characters, or at the right or left side of a character. Consequently, font design will treat this problem by using so-called "flying accents". Flying accents are also called nonspacing accents, because they have a zero width, overlap to the left and can be combined with every base character. Using such a technique reduces a character set in a most dramatic way: every diacritic is needed only twice, once for lowercase and once for uppercase characters. However, this reduction comes at the expense of typographical precision. As base characters differ in width, flying accents will not fit equally well over each of them. Usually, they are constructed in such a way as to fit perfectly over characters of normal width. To achieve a better fit over narrow or broad characters as well, kerning can be built into the font. Kerning is a technique that moves two characters closer together or farther apart; one of the characters, can, of course, be also a diacritic. Kerning works within a single font, not between different fonts. Thus, it is impossible to simply construct a font which has only diacritics and nothing else and still kerns with base characters selected from other fonts. For Latin characters, the kerning tradition varies in different countries. Commercial fonts have from one hundred to one thousand kerning pairs ("Method", the author's OCS font, has nearly three thousand pairs, that is, it has been extensively kerned.)

While the combination of flying accents and kerning is certainly the right solution for OCS, application programs have to be considered. Unfortunately, even major word-processor do not care about the kerning available in fonts and output text without any kerning. Page-layout programs, though, always support kerning. On Windows machines, another problem has surfaced: because flying accents are not used in standard fonts at all, programs and printer drivers may

not expect zero-width characters and may not (yet) handle them correctly. Thus, it is possible that a zero-width character is displayed on-screen as if it had a positive width, resulting in blank space to the right, or flying accents may be visible on-screen, but do not print or are cropped (the combination of Word for Windows 6.0 and the PostScript printer driver for Apple LaserWriters seems to work without problems in this respect).

3.4. Superscripts

For superscripts, in principle the same is true as has been said about diacritics. Raising and lowering of individual characters has always been a feature of word-processors. It is obvious that this approach is too limited to handle a complete superscript alphabet as required for OCS. Consequently, superscript characters have to be implemented as a special font. Again, the same solution can be applied: superscripts can be constructed as flying accents, i.e. zero-width characters which overlap the character written before them. The simple solution that the author of this paper chose is to create a separate font and replace all uppercase characters with the corresponding superscript letter. This makes it very easy for users to enter superscripts, because they do not have to learn any new key combinations. Also, superscripts can be searched for simply as capital letters from a specific font.

4. The Future

4.1. Part I: Encoding

Clearly, the future of encoding is Unicode. Unicode is an encoding scheme with enough room for all the world's characters (most of them, anyway). The Unicode standard is being developed by The Unicode Consortium, founded and supported by several hardware and software manufacturers. For an excellent article about Unicode and its backgound see SHELDON (1991).

The Unicode standard has been published in two thick volumes, the first one being devoted to alphabetic characters, while the second one is devoted exclusively to Asian characters. How does Unicode treat OCS? Basically, Cyrillic OCS is thought of as modern Cyrillic plus x historic characters. It becomes immediately apparent that there are strange elements in this encoding which should never have been encoded as such (for example, izhica with double grave accent). On the other hand, many special characters are missing from this encoding, as any specialist in the field will point out. Glagolitic has not been standardized at all; it is mentioned shortly in the manual as a candidate for future inclusion though.

The current Unicode encoding, then, is not yet the standard expert Slavists are looking for. The good thing about Unicode is, however, that anybody can submit a proposal for the inclusion of new characters to the Unicode Con-

sortium (the rules and guidelines for this are outlined in vol. 1 of the manual). So there is a way out of current limitations, if these suggestions are accepted.

Right now, we are seeing the first steps in implementing Unicode on PCs. Windows 95 is said to support Unicode encoding in (only?) five of its APIs (Application Programming Interface), while Apple says its Unicode support will be "extensive" in "Copland" (System 8), planned for release in 1996, with "true Unicode support in major system components".

Before embracing Unicode as the solution to all encoding problems, however, one should be aware of the consequences Unicode will have. For *complete* Unicode support, all operating systems will have to be rewritten, all current software will have to be replaced with new versions, and, last but not least, Unicode fonts will have to be produced. Also, very powerful PCs will be required to run these Unicode-based operating systems. Hardly any of todays's PCs will be powerful enough or even be in use when full Unicode support finally becomes a reality. It seems realistic to the author to predict that widespread use of Unicode is at least five years away. Clearly, then, Unicode is no solution for today.

4.2. Part II: Technology

Future technology that is emerging now includes QuickDraw GX. QuickDraw GX is a Mac OS extension which takes care of font management, printing etc. It has been introduced as an optional feature with System 7.5 and will be an integral part of the next major Mac OS version which has been announced for 1996.

QuickDraw GX introduces a new font format, TrueType GX fonts. While older fonts (Type 1 and TrueType) continue to work (Type 1 fonts have to be converted first, though), only GX fonts can support new features such as automatic ligatures, automatic context-based allographs, automatic small caps – all in one font. The GX font format has been developed by Apple primarily for better handling of Arabic and similar scripts.

It seems that the new features of GX fonts could also be put to good use in the case of Fraktur fonts, mediæval fonts such as OCS etc. Although the technology of QuickDraw GX fonts seems promising, it has also been called a case of "overengineering". Acceptance of QuickDraw GX by end-users has been slow because of the memory requirements, some bugs in the initial release, and because powerful Macs are needed to perform the necessary calculations. Also, programs have to be rewritten to make use of special GX features, and few programs support GX at present. MS-Word for example, the de-facto standard in the scientific community, does not support special GX features. Adobe, inventors of PostScript Type 1 fonts and publishers of PageMaker, will also not support GX features which they say can be duplicated by using expert Type 1 fonts. The reason is simple: GX fonts are TrueType fonts, and Adobe does not supply TrueType fonts.

Another limitation is that so far QuickDraw GX is tied to the Macintosh and not a cross-platform technology, like other system software components from Apple (QuickTime and QuickDraw 3D are cross-platform, for example). Thus, data transfer between platforms is further complicated by using QuickDraw GX fonts. Also, Fontographer (v. 4.1), the premier font editor on the Mac, does not at present support the generation of TrueType GX fonts.

In spite of all this, Macintosh users still have to reckon with QuickDraw GX, because, as already pointed out, it will be an integral part of the next major Mac OS version.

5. The User

As the present conference proves, many Slavists care about standards. But what about "normal" scientific users who are writing books and papers on their PC and print final output on their laser-printer? What do these users want? From his own experience, the author knows that many users of foreign language fonts do not care about encodings, standards or data exchange with others. They simply want a solution which is convenient, that is, easy for them to use, one which fits in with their localized software (including keyboard driver). This is especially true for Europe with its many different countries and keyboards. Not all users are ready to switch keyboard drivers just to write a word or two in Cyrillic.

In Europe, the urge to move to encoding standards has been much greater within companies whose daily business relies on being able to exchange data between machines and platforms painlessly.

Every current software solution has an installed user base and following. To move these users to a new standard can be a very difficult and time-consuming task, and will always be only partially successful. A radical solution would, of course, be to offer only standard-encoded solutions after such a standard has been arrived at, sacrificing backward-compatibility.

6. Data Exchange

Exchange of data (text files, databases etc.) has previously presented problems for users. As far as Western languages are concerned, these problems are now a thing of the past. Apple bundles a System extension called "PC Exchange" with Macintosh System software; this allows PC disks to be mounted on the Mac desktop, PC files to be dragged to and from the Mac hard-disk etc. so that there is no difficulty in loading PC files into the Mac. Similar utilities now exist for Windows PCs giving those users access to Mac disks and files. Allowing PC disks and files to be manipulated as if they were Mac disks and files does not change anything in the content of the files, though, that is, so far no file conversion takes place. Fortunately, most cross-platform Mac programs directly read in texts made with their counterpart on Windows PCs and vice versa, re-encoding

all characters as required on the fly. Also, there are data transfer utilities with filters for hundreds of application combinations.

If, however, Slavic or Baltic languages, which have a different encoding from Western languages, have been used in texts, these filters and conversion procedures render complete nonsense because they treat all characters as if they were from the standard Western encoding scheme. This is also true of OCS texts written using today's OCS fonts. Consequently, other means of moving OCS text intact from a PC into the Mac and back again have to be found.

Fortunately, a solution is available on the Macintosh. Apple has shipped a program called Apple File Exchange (AFE) with System software 7.1. This is an application program which serves a similar purpose as the control panel PC Exchange which ships with System 7.5 and supersedes AFE. AFE reads PC disks and allows file transfer from a PC disk to the Mac hard-disk or vice versa. This in itself is, as we have seen, not enough. There is, however, a little-known add-on to AFE, called Text-to-Text Converter. This gem converts character codes on the fly while copying a text file, and, best of all, these translation tables are user-definable. They simply consist of two lists of corresponding ASCII or hex values. If the program encounters a character having the first value, it replaces it with a character having the second value. Furthermore, not only one-to-one replacements are possible, but also one-to-many replacements. This feature can be used, for example, to flag problems in the conversion process which could later be worked on by using a second conversion table or by a search-and-replace operation from within a word processor.

The author of this paper has already created translation modules for Cyrillic between Windows and Macs, and for Latin Eastern Europe (EE encoding to CE encoding). A conversion module for corresponding OCS fonts would be possible, if users required it.

There are, however, two limitations connected with this approach. The first one is that it works on plain ASCII text only. As this includes RTF files (Rich Text Format) which preserve character and document formatting, this really is no problem. The RTF format is supported by most major word processors, making this an ideal means of exchanging texts between platforms and programs. The second limitation lies in the fact that these converters do not work context-sensitively; thus, it is impossible to translate only selected words in a mixed-language text. Again, this does not present a problem if the source text is OCS only.

7. Conclusion

If we ask ourselves what can be achieved today in terms of standardization, the author would reach the following conclusion:

Possible and practical today are platform-dependent standards, that is one standard for Windows machines and one standard for Macintoshes. Utilities

such as AFE (Apple File Exchange) with its Text-to-Text converter can be used for data exchange between platforms. The author of this paper is convinced that these platform-dependent standards will be needed at least for several years to come before a common Unicode-based standard can gradually replace them. There can be no doubt, however, about the fact that not all current or future users care about standards or data exchange. They want the most convenient solution for their individual working habits, their local keyboard and software, and they want it now.

8. References

The Unicode Consortium:
1991-92 The Unicode Standard. Worldwide Character Encoding. Version 1.0. Vols. 1-2. Addison-Wesley: Reading, MA.
Apple Computer:
1995 Inside Macintosh: QuickDraw GX Printing. Cupertino.
Apple Computer
1995 QuickDraw GX Font Formats and The TrueType Font Format Specification. Cupertino.
Sheldon, K.M.:
1991 ASCII goes global. *Byte* 7, 108–116.

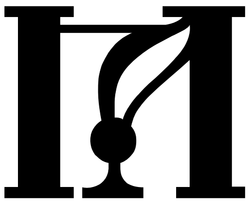
Sebastian Kempgen University of Bamberg/Germany

Appendix

Original overhead slides

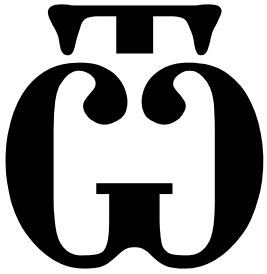
Outline

- АБВАЖабгдѵжѣѳю а́ъ̀а̀
- ѦБѮꙊѬѰѮѴ И҃С И́И҃С ѦЄѪ
- ABVGDEZIJKLMNOPQ
- ⰘⰞⰞⰨⰑⰟⰅⰇⰗⰇⰖⰆ

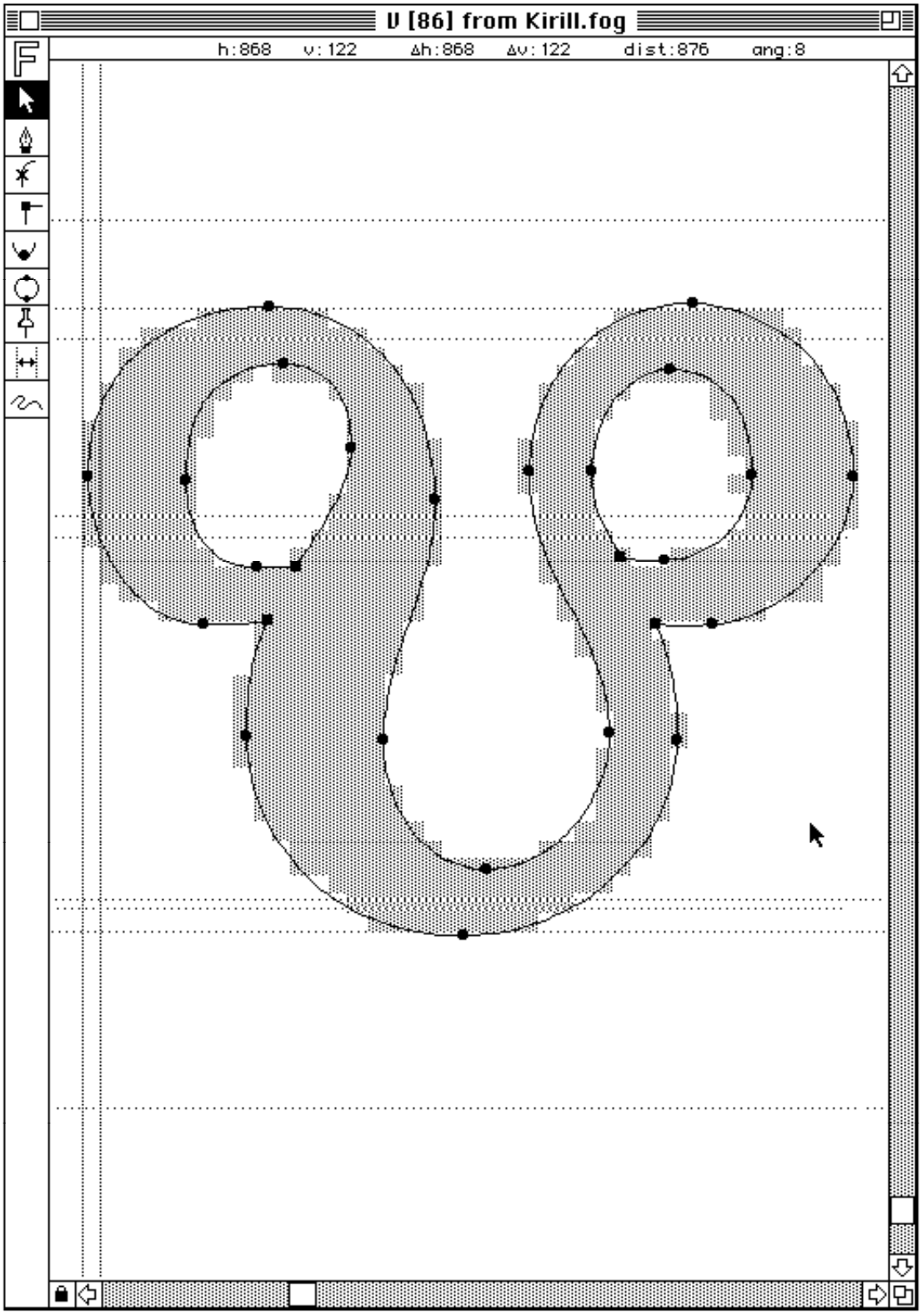


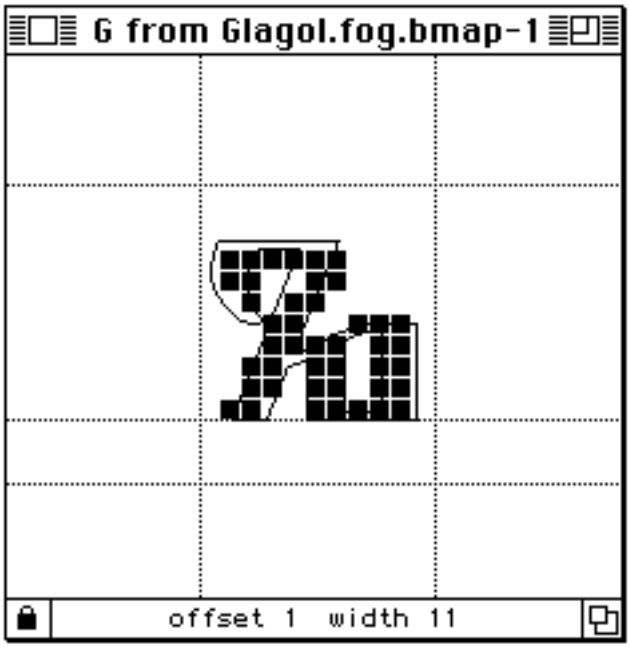
OCS – The Situation

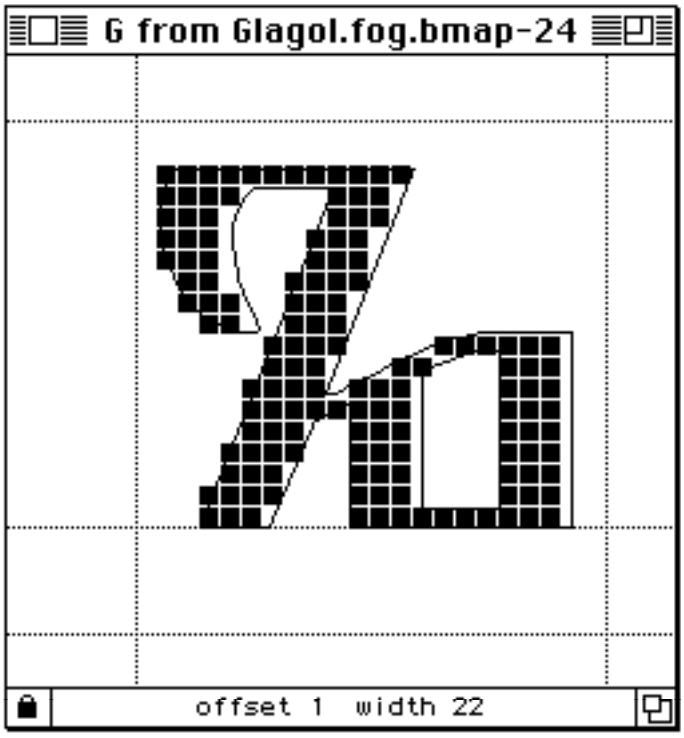
- No normative alphabet/orthography ѡ, оу
- Graphemes and allographs ч/ꙗ, ѭ/ѧ ѧ/ѧ/ѧ
- Many diacritics ѝ й ѝ й й ѝ й
- Superscript alphabet $^{\text{х}}$й
- Many ligatures ш̅ пц ѣ ѣ



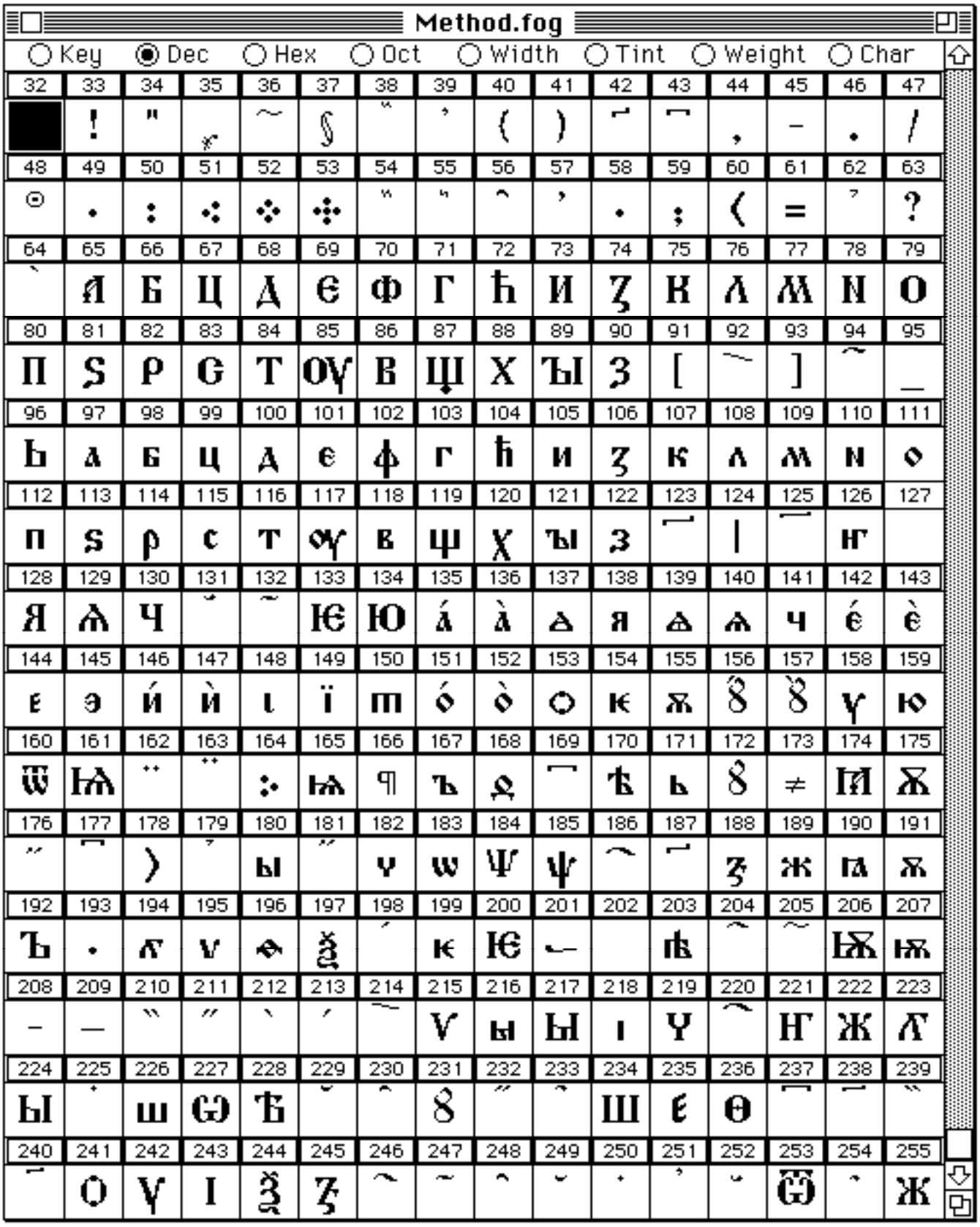
Sebastian Kempgen **Blagoevgrad, July '95**



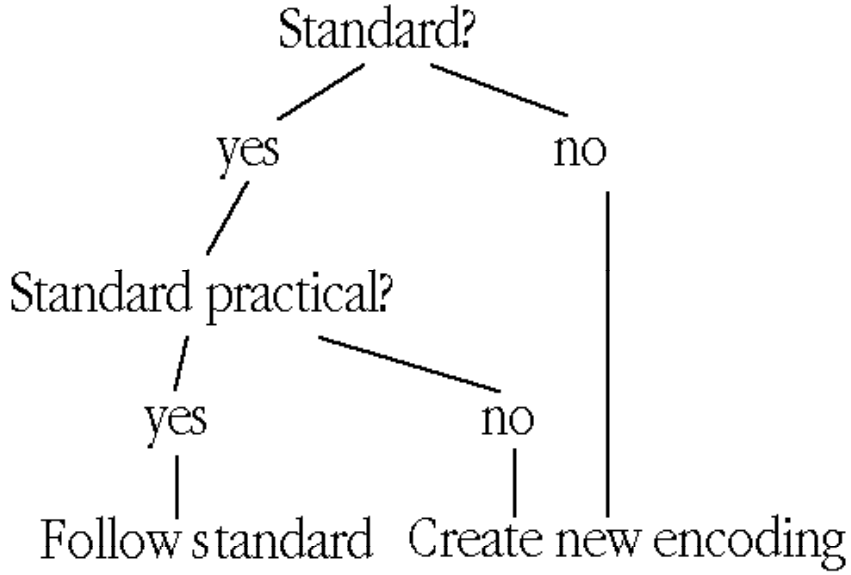

G from Glagol.fog.bmap-1
offset 1 width 11


G from Glagol.fog.bmap-24
offset 1 width 22

Sebastian Kempgen **Blagoevgrad, July '95**



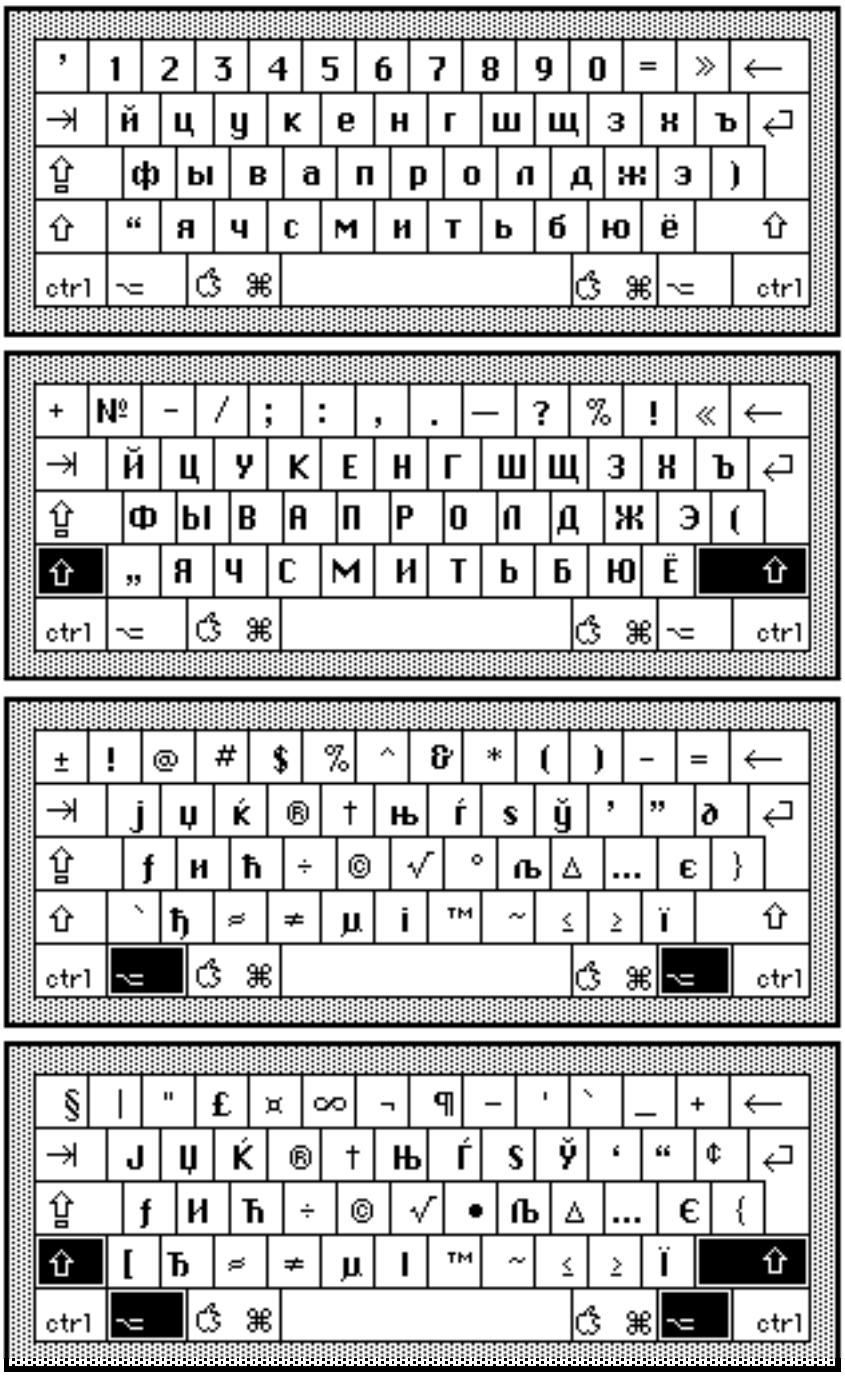
Encoding Strategies



- Vowel on V, cons. on C: **О > O, Р > П**
- Character pairs **Œ œ > Ѫѫ**
- Basic ABC = Lower ASCII **ЯБЂГДЄ**
- Phonetic/graphic similarity **Z > З**
- Local keyboard driver **Ä, Ü > Я, Ю**

Sebastian Kempgen Blagoevgrad, July '95

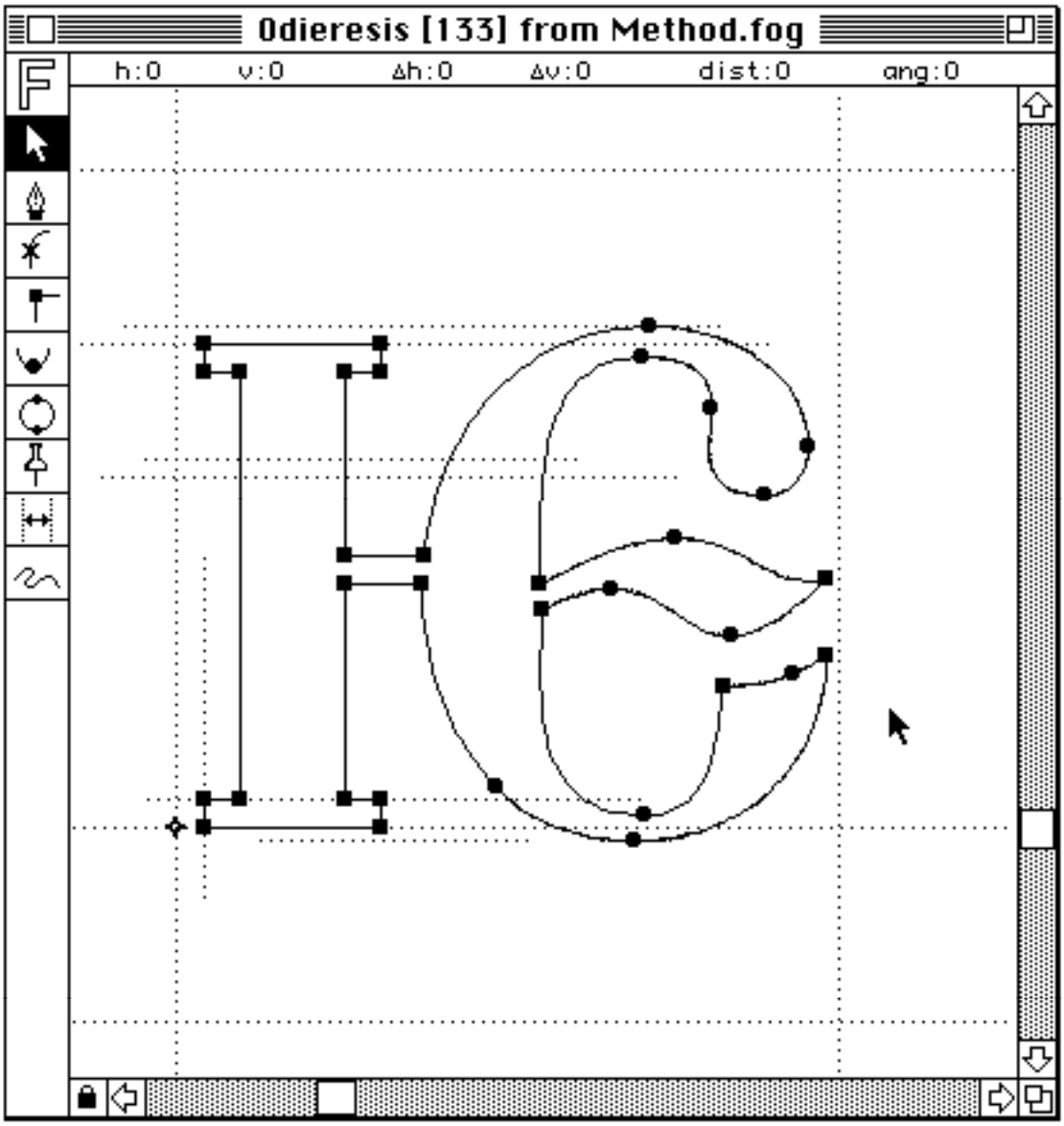
Macintosh Keyboard



(Ukrainian Mac KB)

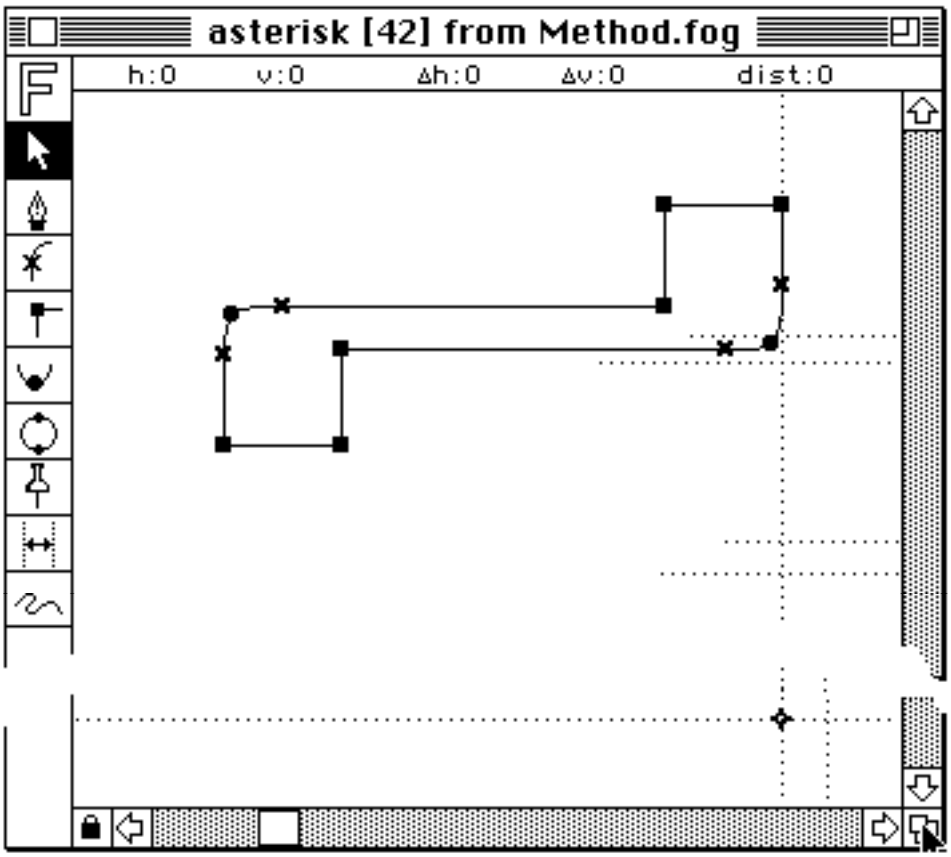
Sebastian Kempgen Blagoevgrad, July ’95

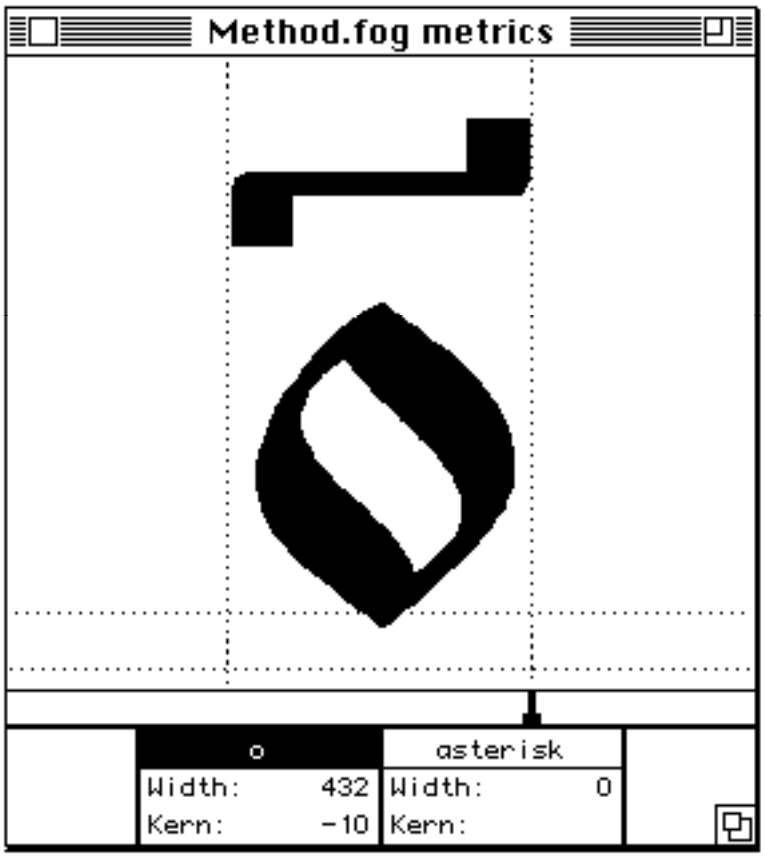
Character Outlines



Ö

Diacritics, "Flying Accents"





Superscripts

what you enter >	*what you see*
gpS^´ >	**гпⷭ҇ь**
srDce >	**срⷣце**

иⷯ

Unicode

- New system software versions
- New application software versions
- New fonts
- New, powerful PCs

- Technology for the next century

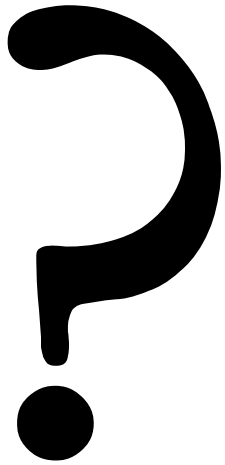
QuickDraw GX

- Automatic lig., allographs, SMALL CAPS
- Arabic

- Slow acceptance by users
- Slow support by software vendors
- Not a cross-platform technology
- No font editors

- Built into "Copland" (Mac OS 1996)

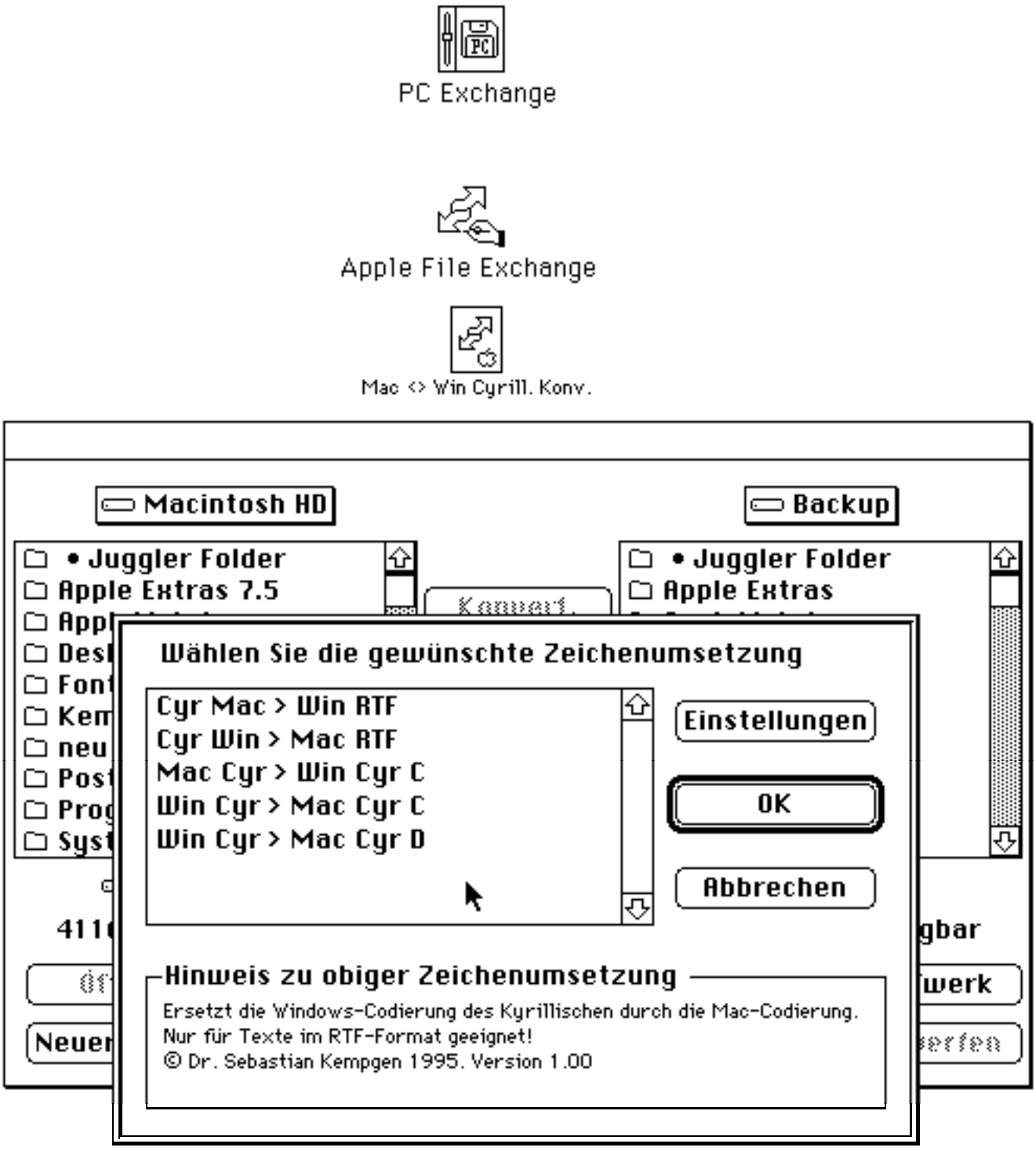
The User



- Convenient
- For today's PCs

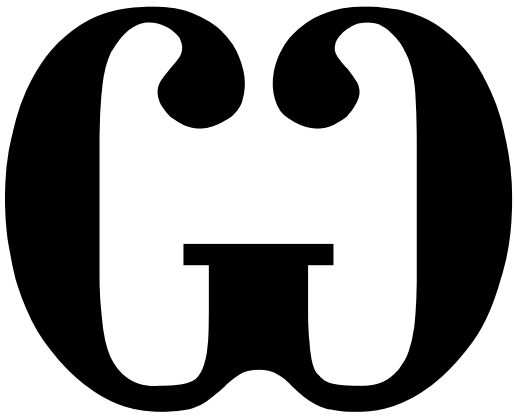
Sebastian Kempgen Blagoevgrad, July '95

Data Exchange



Conclusion

- Platform-dependent standards
- Mac, Windows
- Utilities for data exchange



Bibliographische Angaben / Bibliographical Entry:

Sebastian Kempgen: Complex Script Systems on Today's Personal Computers. In: D. Birnbaum, A. Bojadzhiev, M. Dobreva, A. Miltenova (eds.), *Computer Processing of Medieval Slavic Manuscripts, Proceedings, First International Conference, 24–28 July, 1995, Blagoevgrad, Bulgaria*. Sofija 1995, 68–78.

February 2021, postprint (incl. original overhead slides as an appendix)