UNICODE FOR SLAVISTS
or
HOW TO SLICE A PINEAPPLE

Paper presented to the colloquium “Стандардизација Старословенског Ћириличког Писма и Његова Регистрација у Уникоду”, Belgrade, 15-17 October 2007
by
Ralph Cleminson

From its inception, or rather since its progressive incorporation into various computer applications, the Unicode standard has become a vital part of computer use, particularly for those of us who use more than one writing system. Indeed, since Unicode is an integral part of the XML Recommendation, anyone involved in creating or using electronic text is almost certainly also using Unicode. Most of us can probably remember the days, not so long ago, when everything apart from the ninety-six characters comprising the Latin alphabet, Arabic numerals and selection of punctuation and symbols included in the ASCII standard (ISO-646) were distributed amongst various different code-pages, with the result that one user’s Greek text could appear as a series of Hebrew characters on another user’s computer, because the code-points used for Greek characters on the sender’s system were the same as those used for Hebrew on the recipient’s. To prevent this, a unified standard was developed, according to which each code-point should represent one and one only of the characters of the world’s writing systems, and each character should likewise be represented by only one code-point. This is the purpose of Unicode, and its remarkable success can be measured by the fact that the pages full of inappropriate characters that one frequently used to encounter are largely a thing of the past.

Technically, Unicode is two standards. In the late 1980s and early 1990s, efforts towards a single standard for character encoding were being made both by the Unicode Consortium and within the International Organisation for Standardisation, and their first proposals were made independently of each other. It was recognised by both organisations that a unification of the two was desirable, and this was achieved in 1993 with the publication of Unicode 1.1 and ISO 10646-1, which are identical in content. Since then the two standards have been synchronised, so that every subsequent version has been exactly the same in both, though formally they are maintained separately and have their own administrative procedures.

The starting-point from which Unicode was developed was a number of earlier national and industry standards covering various ranges of characters. These were subsumed into Unicode with provision for retroconversion, which meant that anything encoded in Unicode 1.1 could be encoded in one of the foundation standards, and vice versa. In some cases this necessitated the inclusion in Unicode of “compatibility characters” (defined as “those that would not have been encoded (except for compatibility) because they are in some sense variants of characters that have already been coded.”) The consequences of this are most evident in the treatment of Far Eastern writing systems. As far as European scripts are concerned, they chiefly manifest themselves in the encoding of accented characters. Thus Latin á may be represented either as U+00E1 or as U+0061+0301, and Greek ά as either U+1F71 or U+03B1+0301, and these alternative encodings are explicitly designated as *canonically equivalent* in the standard.

Cyrillic а́, on the other hand, may only be encoded as U+0430+0301, because there was no such precomposed character in the earlier standards upon which Unicode was based. As far as possible Unicode avoids ambiguities and equivalences, and there is a strong policy not to admit any more precomposed characters to the standard.

In practice the existing precomposed characters may be exploited for typographical convenience, but it should always be remembered that it is not actually the function of Unicode to provide this. Encoding issues and typographical issues are separate, and Unicode is concerned only with the former.

The four pillars of text encoding are structure, portability, structure and multiple use[*], and it is the purpose of Unicode to allow plain text to be encoded in fulfilment of these aims. It is not the purpose of Unicode to record the visual representation of everything that can be represented, an aim that would be as inconvenient in practice and as unfeasible in theory as a 1:1 scale map[†]. For this reason Unicode characters "are represented by code points that reside only in memory representation, as strings in memory, on disk, or in data transmission"–that is to say, not in any form recognisable to a human being–and are distinguished by their systemic or semantic features. For this reason Latin A, Greek Α and Cyrillic А are recognised as separate characters even though they may appear identical in print, because it makes a difference which is which–most obviously for sorting and searching operations. This principle may equally apply within scripts: the Icelandic letter Ð (U+00D0) is not the same character as the Croatian letter Đ (U+0110), if only because the one corresponds to lower-case ð and the other to lower-case đ.

Similarly, in the conventions of modern Church Slavonic typography certain words *must* be written with superscript letters. Thus the canonical spelling is **бⷣца; богородица** is a non-canonical but possible spelling (which may be used, for example, in musical texts), while **бдца** is a mistake. The distinction between **д** and superscript **д** is therefore significant within this system, and the two are not interchangeable. In recognition of this a series of Cyrillic superscript characters has recently been approved by the Unicode technical committee, and will shortly appear within the standard. (It should be noted that this distinction does not exist in all systems of Cyrillic writing, and in most mediæval manuscripts it would make no difference at all whether the scribe wrote **прѣдꙗви** or **прѣⷣꙗви**. One could debate whether in such a case the superscript was better indicated by the use of a distinct character or by mark-up.)

In the same way, where visual differences have no bearing on the meaning of the text, they are not recognised by Unicode. Thus, while the form of the lower-case Cyrillic letter т is notably different in a roman typeface, a Russian italic typeface, and a Serbian italic typeface, it is only one character. From the point of view of *plain text*–text as a string of characters, which it is the business of Unicode to encode–it matters not at all whether the text is roman or italic, Russian or Serbian. From other points of view, of course, it matters a great deal, and therefore there are mechanisms, in the form of mark-

[*] These were enunciated as long ago as 1995 in David Birnbaum, "How Slavic Philologists Should Use Computers", *Computer Processing of Medieval Slavic Manuscripts: Proceedings [of the] First International Conference, 24–28 July, 1995, Blagoevgrad, Bulgaria,* Sofia: Marin Drinov, 1995, pp.19-28, an article which is still worth perusing.

[†] For the theoretical impossibility of the latter, see Umberto Eco, "On the impossibility of drawing a map of the empire on a scale of 1 to 1", *How to travel with a salmon*, New York: Harcourt Brace, 1994, pp.95–106.

up, etc., to indicate it, but these are not part of Unicode: they work alongside it and with it.

How then does Unicode fit in to the wider operation of text encoding? If you are working with in an entirely closed system, there is no need to use Unicode at all–you can represent characters however you like. This however is not recommended unless you are absolutely certain that no document you produce or any part of it will ever be used by anyone else, ever be needed for any other purpose, or ever be required beyond the life expectancy of the hardware and software on which you produce it. This is not a usual situation.

One may assume, therefore, that Unicode will be used, and that sooner or later the need may arise to provide for characters which are not in Unicode, either because they ought to be in it but have not yet been included, or because they do not fulfil the criteria for inclusion and therefore never will be. In the first case, Slavonic mediævalists are fortunate that a proposal including the mediæval Cyrillic characters that had been identified as missing from Unicode was accepted by the Unicode Technical Committee in February of this year. The accepted proposal can be seen at http://std.dkuug.dk/jtc1/sc2/wg2/docs/n3194.pdf; it also includes the superscript letters mentioned above, as well as characters required for modern non-Slavonic languages. Because of the time taken by administrative procedures, although it is now certain that they will form part of the standard, it is unlikely that this will be formally accomplished until next year. In the mean time caution should be exercised, as though it is probable, it is not absolutely certain that the characters will be assigned to the code-points at which they appear in the proposal.

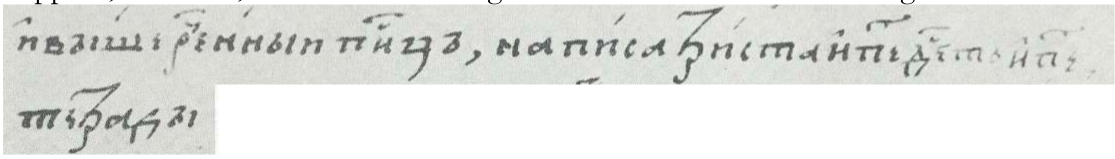
For the rest, there is the Private Use Area, a range of characters which exist within the standard but are undefined within it and may be used for any purpose a user or group of users chooses. There are 6,400 such code-points within the Basic Multilingual Plane (specifically, U+E000-F8FF), which should be enough for most purposes (at least when dealing with European writing systems), but should that prove insufficient, Planes 15 and 16 are also allotted as supplementary private use areas, providing an additional 131,068 code-points. Any character that one is using which is not already provided for by Unicode can thus be assigned to a code-point in the Private Use Area, and the adoption of Unicode does not, therefore, in any way limit the range of characters that one can use.

The Private Use Area is, by definition, private, that is to say non-standard, and it is to say the least unlikely that two users will choose independently to put the same character at the same code-point. Relying *only* on the Private Use Area would therefore re-create the problems of the pre-Unicode period. There are, however, ways of getting round this which allow us to have the best of both worlds. It is done simply by using mark-up to indicate to other users how the Private Use Area has been used.

This is best illustrated from an actual example. In this example I shall use the mark-up scheme devised by the Text Encoding Initiative, a project which has been developing models for the electronic encoding of texts since 1988 and has now reached a considerable level of sophistication[*]. This has the advantage of being highly developed and of being familiar to many users. One can of course achieve the same ends using other schemes of mark-up if these are more appropriate to the purpose being pursued.

[*] Details of the TEI can be found on its website, http://www.tei-c.org/, which includes very extensive documentation.

Suppose, therefore, that one is encoding a text that includes the following:



and it is desired to preserve the orthographical peculiarities of the original. Provided that you have the necessary glyphs in the Private Use Area of your font, you can immediately type

и҆ вы́шєрⷱєнныи пиⷭцъ напи́са трі́ста и҆ пєⷮдⷭєть и҆ пєⷮ тєтрады

in the virtual certainty that nobody else will be able to read it. It is better, therefore, to start the encoding using exclusively Unicode characters*:

и҆ вы́шєрч҆є́нныи пиⷭцъ напи́са три́ста и҆ пєтдⷭєть и҆ пєт тєтрады

One can then add mark-up to indicate the ligatures

и҆ вы́шєрч҆є́нныи пиⷭцъ напи́са <g ref="**#trlig**">**тр**</g>**и́ста и҆ пєтдⷭєть**
и҆ пєт тє<g ref="**#trlig**">**тр**</g>**ады**

and one can do the same for the three-legged **ᲅ**:

и҆ вы́шєрч҆є́нныи пиⷭцъ напи́са <g ref="**#trlig**">**тр**</g>**и́с**<g
ref="#t3">**ᲅ**</g>**а и҆ пєтдⷭє**<g ref="#t3">**ᲅ**</g>**ь и҆ пєт** <g
ref="#t3">**ᲅ**</g>**є**<g ref="**#trlig**">**тр**</g>**ады**
<g ref="**#t3**">

The non-Unicode alternatives are indicated by enclosing the characters which they are to replace within a <g> element with an attribute indicating what it refers to. These are referenced elsewhere in the document:

```
<char xml:id="trlig">
<charName>CYRILLIC TVRDO-RCI LIGATURE</charName>
<charProp>
<localName>entity</localName>
<value>trlig</value>
</charProp>
<mapping type="PUA">U+F474</mapping>
</char>
```

The reference includes both an indication of where the character resides in the Private Use Area, so that it can be processed on one's own system, and an explicit definition in a human language for the benefit of other users who might otherwise have no means of knowing what the character was.

We can then go on to indicate superscripts:

и҆ вы́шєр<hi rend="**sup**">**ч҆**</hi>**є́нныи пи**<hi rend="**sup**">**ⷭ**</hi>**цъ напи́са**
<g ref="**#trlig**">**тр**</g>**и́с**<g ref="#t3">**ᲅ**</g>**а и҆ пє**<hi
rend="**sup**">**ᲅ**</hi>**д**<hi rend="**sup**">**ⷭ**</hi>**є**<g ref="#t3">**ᲅ**</g>**ь и҆ пє**<hi
rend="**sup**">**ᲅ**</hi> <g ref="#t3">**ᲅ**</g>**є**<g ref="**#trlig**">**тр**</g>**ады**

Finally, we might decide that it is useful to be able to supply the full content of abbreviations:

и҆ вы́шєр<choice><ex>**є**</ex><seg
type="**sup**">**ч**<am>**҇**</am></seg></choice>**є́нныи**
пи<choice><seg type="**sup**">**с**<am>**҇**</am></seg><ex>**ь**</ex></choice>**цъ**
напи́са <g ref="**#trlig**">**тр**</g>**и́с**<g ref="#t3">**ᲅ**</g>**а и҆ пє**<seg
type="**sup**">**ᲅ**</seg>**д**<choice><ex>**є**</ex><seg

[*] One could in fact do it the other way round, starting with the quasi-facsimile transcription and transforming this into Unicode, but this is a much more complicated operation and provides a much less clear illustration of the principles and methods involved.

```
type="sup">с<am>҇</am></seg></choice>є<g ref="#t3">т</g>ь и҆ пє<seg
type="sup">т</seg> <g ref="#t3">т</g>є<g ref="#trlig">тр</g>ады
```

Putting it all together, we produce the following document:

```
<!DOCTYPE TEI PUBLIC "-//TEI P5//DTD Main Document Type//EN"
"http://www.tei-c.org/release/xml/tei/schema/dtd/tei.dtd"[
      <!ENTITY % TEI.header 'INCLUDE' >
      <!ENTITY % TEI.core 'INCLUDE' >
      <!ENTITY % TEI.textstructure 'INCLUDE'>
      <!ENTITY % TEI.gaiji 'INCLUDE'>
      <!ENTITY % TEI.transcr 'INCLUDE'>
      <!ENTITY % TEI.linking 'INCLUDE'> ]>
<TEI>
<teiHeader>
<fileDesc>
<titleStmt>
<title>An example of transcription</title>
</titleStmt>
<publicationStmt>
<p>An unpublished document </p>
</publicationStmt>
<sourceDesc>
<p> Hilandar MS 444, f.360v.</p>
</sourceDesc>
</fileDesc>
<encodingDesc>
<charDecl>
<char xml:id="trlig">
<charName>CYRILLIC TVRDO-RCI LIGATURE</charName>
<charProp>
<localName>entity</localName>
<value>trlig</value>
</charProp>
<mapping type="PUA">U+F474</mapping>
</char>
<glyph xml:id="t3">
<glyphName>CYRILLIC THREE-LEGGED TVRDO</glyphName>
<charProp>
<localName>entity</localName>
<value>t3</value>
</charProp>
<mapping type="PUA">U+F471</mapping>
</glyph>
</charDecl>
</encodingDesc>
</teiHeader>
<text>
<body>
<!-- Text goes here. --><p> и҆ вы́шєр<choice><ex>є</ex><seg
type="sup">у<am>҇</am></seg></choice>єн́ныи
 пи<choice><seg type="sup">с<am>҇</am></seg><ex>ь</ex></choice>цъ
 напи́са <g ref="#trlig">тр</g>и́с<g ref="#t3">т</g>а и҆ пє<seg
type="sup">т</seg>д<choice><ex>є</ex><seg
type="sup">с<am>҇</am></seg></choice>є<g ref="#t3">т</g>ь и҆ пє<seg
type="sup">т</seg> <g ref="#t3">т</g>є<g ref="#trlig">тр</g>ады
</p><!-- More text goes here. -->
</body>
</text>
</TEI>
```

The important thing to note is that this transcription contains *all the information* that was in the transcription made simply by typing using characters from the Private Use Area, but in a form that will not be lost in transmission or transference to another platform. The document is of course designed to be read by the computer, not by the human eye. While it is perfectly possible to view the text directly from this document, in practice it is unlikely that one would wish to do so. The initial encoding is normally a starting-point for transformations that present the information in a form in which it is required for a particular purpose.

One could, for example, insert the actual glyphs for the non-Unicode characters into the text in place of the more descriptive encoding in the document. This can be done with XSLT using the following file.

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
  <xsl:output method="xml" indent="yes" encoding="ISO-8859-1" />
<xsl:template match="@*|*|processing-instruction()|comment()">
<xsl:copy>
  <xsl:apply-templates
select="*|@*|text()|processing-instruction()|comment()" />
  </xsl:copy>
  </xsl:template>
<xsl:template match="g">
<xsl:variable name="foo">
  <xsl:value-of select="substring(@ref,2)" />
  </xsl:variable>
  <xsl:text disable-output-
escaping="yes">&amp;#x</xsl:text><xsl:value-of
select="substring(//*[@xml:id=$foo]/mapping,3)" />;
  </xsl:template>
  </xsl:stylesheet>
```

It will be noticed that this transformation uses the code-points for the glyphs declared in the header of the actual document, so that provided they are declared correctly at this one point, every instance of the characters in the actual text will be correctly transformed. There is–and this is typical–a loss of information between the input and the output, insofar as the information about these characters in the output file is reduced to the code-points for Private Use Area glyphs, but this does not matter because the input file continues to exist, along with all the information in it. Since electronic encodings may have multiple uses, it need not be the case that all the information in them is required for each of the uses to which they are put.

In isolation this particular transformation may not be especially useful, but one can see that in combination with others it could be of considerable use, for example in producing a quasi-facsimile view of the text. A more complete kind of transformation would be provided by the following script:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
                version="2.0" >

<xsl:output method="html" indent="yes" encoding="ascii" />
 <xsl:template match="/">
```

```
<html xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xsl:version="1.0">
 <head>
   <title>Transcription</title>
 </head>
 <body>
   <xsl:apply-templates/>
 </body>
</html>
 </xsl:template>

 <xsl:template match="teiHeader"/>
 <xsl:template match="//p">
  <p>   <xsl:apply-templates/></p>
 </xsl:template>

 <xsl:template match="//*[@type='sup']"><em><xsl:apply-
templates/></em></xsl:template>

 <xsl:template match="ex">(<xsl:value-of select="."/>)</xsl:template>

 <xsl:template match="am"></xsl:template>

</xsl:stylesheet>
```

This will generate an HTML document containing a transcription of the text into modern Cyrillic, thus:
и̓ вышер(е)*ч*ен́ныи пи*с*(ь)цъ напи́са три́ста и̓ пе*т*д(е)*с*еть и̓ пе*т* тетрады.

These two examples give a very limited glimpse of how a source document can serve as the basis for transformations, and of the use of Unicode within the source document. It must be remembered that the initial encoding is very rarely the same as the final output, and that in particular documents designed as "published" work, whether on paper or on the screen, are usually the result of one or more transformations using Formatting Objects, LaTeX or other applications designed to provide the typographical polish requisite for this sort of document. If а̓́ is encoded as `&#x0430;&#0313x;&#x0301;`, this information is securely and unambiguously available to any system; an XML browser probably will display it, and the fact that it will probably not display it very well is not a defect, as this is not what it was designed for: there are other applications for this. In particular it should be noted that Unicode is *not restrictive* in its application to text encoding; on the contrary it is *permissive* in allowing the encoding of any string of characters in a way that will preserve the information across platforms and media. In the words of the great English applied mathematician Charles Babbage (1791-1871), "Propose to an Englishman any principle, or any instrument, however admirable, and you will observe that the whole effort of the English mind is directed to find a difficulty, a defect, or an impossibility in it. If you speak to him of a machine for peeling a potato, he will pronounce it impossible: if you peel a potato with it before his eyes, he will declare it useless, because it will not slice a pineapple."[*] Unicode does very well what it is intended to do, and there are other machines for slicing the pineapples.

[*] In fairness to Babbage, and to the English nation, it should be remembered that this famous utterance was made after the failure of a funding application.