Sonderzeichen, TEI und Unicode

TEI-Guidelines Kap. 5

Behandlung von Sonderzeichen

- Bei Transkriptionen älterer und/oder handschriftlicher Texte häufig Sonderzeichen
- Inzwischen zahlreiche Sonderzeichen im Unicode-Standard definiert
- Außerdem bietet die TEI im gaiji-Modul Elemente an, durch die Sonderzeichen definiert, beschrieben und in der Transkription eingesetzt werden können

Was ist Unicode?

- „Internationaler Standard, in dem langfristig für jedes sinntragende Schriftzeichen oder Textelement aller bekannten Schriftkulturen und Zeichensysteme ein digitaler Code festgelegt **wird**.“ (http://de.wikipedia.org/wiki/Unicode)
- Bzw. „festgelegt **werden soll**.“ (OD)
- Unicode ist **kein** Zeichensatz / font (wie Arial, Courier o.ä.), sondern definiert die digitale Kodierung von Zeichen, unabhängig davon, ob diese in einem bestimmten Zeichensatz dargestellt werden können.

Warum Unicode?

- Ältere Standards der Zeichencodierungen konnten lediglich 128 (ASCII, 7 bit) oder 256 (z.B. ISO-8859, 8 bit) Zeichen codieren
- Folge: für unterschiedliche Schriftsysteme mussten verschiedene Zeichencodierungen entwickelt werden und ggf. angegeben werden, in welcher Zeichencodierung eine Datei gespeichert ist (z.B. ISO-8859-1, ISO-8859-5 usw.)

Warum Unicode?

- Unicode soll die verschiedenen miteinander inkompatiblen Zeichenkodierungen ersetzen
- In Unicode 1.0 sollten alle Schriftzeichen der Welt durch 65.536 (2^{16}) sog. „codepoints“ repräsentiert werden
- Inzwischen erweitert auf 17 Bereiche („planes“) von je 65.536 codepoints -> 1.114.112 mögliche Zeichen
- Codepoints werden hexadezimal angegeben: z.B. U+0041 = A

Exkurs Hexadezimalsystem

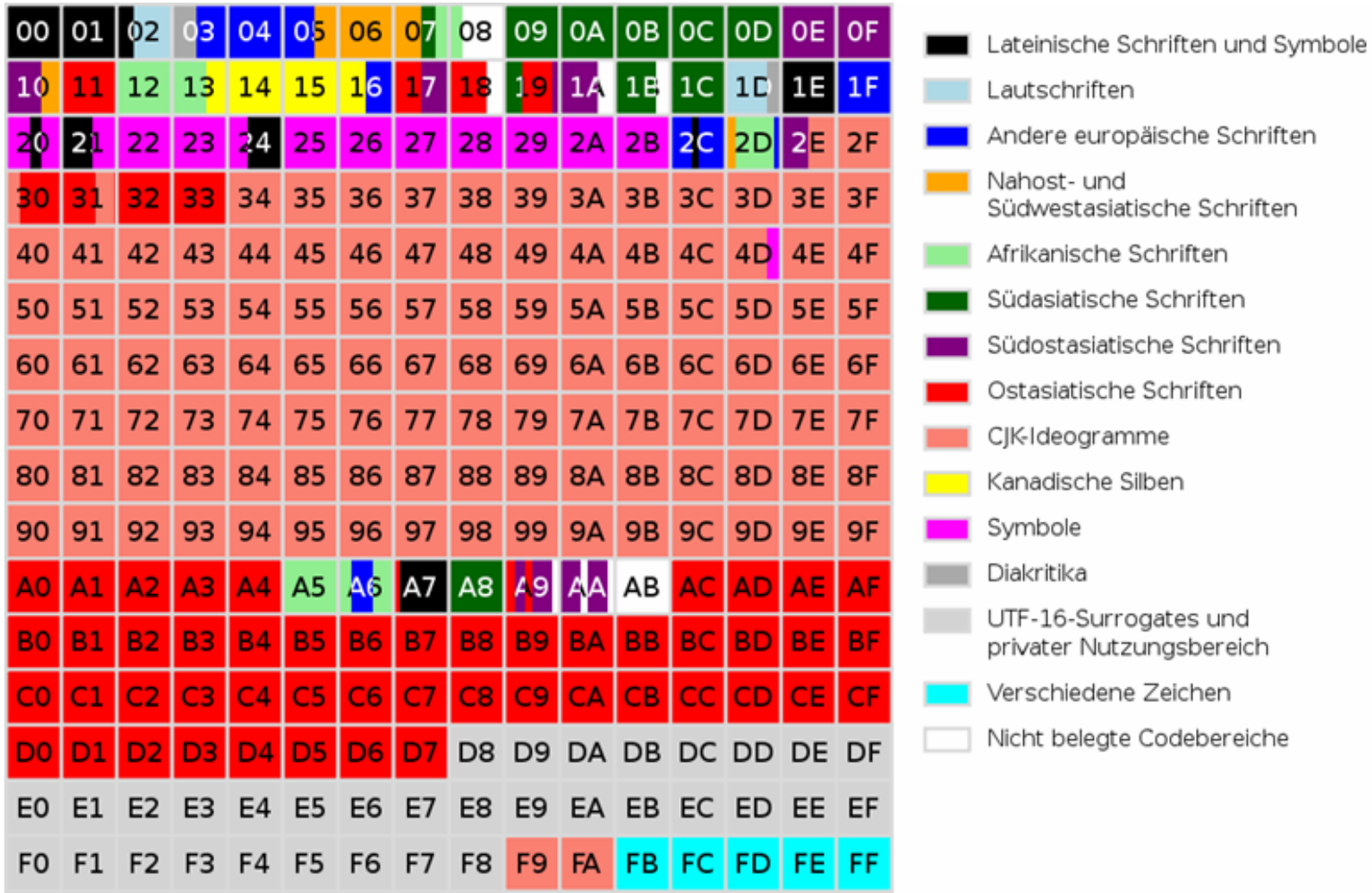
- 16 Ziffern: 0-9, A-F
- A = 10, B=11 usw.
- Hexadezimal 10 = Dezimal 16
- Hexadezimal 1A = Dezimal 26
- Hexadezimal 1B = Dezimal 27
- Hexadezimal 20 = Dezimal 32
- Hexadezimal FF = Dezimal 255
- Usw.

Warum Unicode?

- Unicode-Standard wird ständig durch das „Unicode Consortium“ kontinuierlich weiterentwickelt
- Aktuelle Version ist Unicode 6.2 (Sept. 2012)
- Lateinisch, Griechisch, Kyrillisch, Arabisch, Hebräisch, CJK
- Aber auch so Schriften wie Balinesisch, Gotisch, Glagolitisch, Ogham, Linear B usw.
- Mehrere „Private Use Areas“ (PUA)
- Ergänzungswünsche können (und sollten) dem Unicode Consortium gemeldet werden

Was gibt es in Unicode?

- „Normale“ Schriftzeichen: a b c δ Д ى א ण
- Satzzeichen „ “ ? ! ,
- Whitespace
- Combining Diacritical Marks: ◌̈ ◌̎ ◌̋
- Vorkombinierte Zeichen á ä ť ů ﭖ ḗ ḝ
- Symbole ☭ ♘ ♂ ☿ ∑
- Steuerzeichen Wagenrücklauf, EOF
- …



Quelle: http://de.wikipedia.org/wiki/Datei:Roadmap_to_Unicode_BMP_de.svg

Wie finde ich das Zeichen, das ich brauche?

- Codecharts unter www.unicode.org/charts/
- Datenbank unter www.decodeunicode.org
- Aktueller: www.fileformat.info/
- Oder www.isthisthingon.org/unicode/index.php
  (The UniSearcher)

Was ist ein "Encoding"

- Unicode ist lediglich ein abstrakter Standard, der jedem vorhandenen Zeichen eine Nummer (den codepoint) zuweist
- Codepoints werden hexadezimal angegeben (U+1F46 usw.)
- Die Codierung (encoding) legt fest, in welcher Form die codepoints in einer Datei gespeichert werden
  - UTF8, UTF16 usw. sind **nicht gleichbedeutend mit Unicode**, sondern Standards, wie Unicode-Zeichen gespeichert werden
- UTF-16 -> jedes Zeichen wird mit 2 Byte gespeichert (entspricht dem Codepoint)
  - Big-Endian (höherwertige Bits zuerst), Little-Endian (niedrigere Bits zuerst)
- UTF-8 -> Häufige Zeichen (lateinisches Alphabet) werden in 1 Byte gespeichert, seltenere in 2 oder 3 Byte
  - D.h. bei einem Text, der nur aus lateinischen Buchstaben ohne Umlaute besteht, ist eine UTF-8-codierte Datei nur halb so groß, wie eine UTF-16

Codierung in XML angeben

- Encoding wird in der XML-Declaration angegeben
  - <?xml version="1.0" **encoding="UTF-8"** ?>
- Sollte immer (!), auch in XSLT-Skripten usw. angegeben werden

- BOM = Byte Order Mark
  - Markierung zu Beginn einer Datei (2-4 Byte), die das encoding angibt
  - Z.B.
    - UTF-8: EF BB BF
    - UTF-16 (BE) FE FF
    - UTF-16 (LE) FF FE
  - Führt oft zu Problemen, wenn ein XML-Editor automatisch eine BOM schreibt, ein anderer sie aber ignoriert. Dann tauchen vor der XML-Deklaration komische Zeichen auf: ï»¿, þÿ, ÿþ
  - Lösung: Einstellungen der XML-Editoren prüfen, Zeichen löschen

Kodierung von Unicode in XML-Dateien

- Entweder Zeichen direkt einfügen, z.B. mit Oxygen:

  α intuitiv lesbar, wird aber – je nach Zeichensatz – nicht angezeigt
  außerdem Verwechselungen bei 'ähnlichen' Zeichen möglich

- Oder mit Zeichenentitäten:

  Hexadezimal: `&#x0364;` gut, entspricht dem Codepoint

  Dezimal: `&#945;` bitte nicht!

  Benannt: `&alpha;` oft besser lesbar, aber die Entitätsnamen müssen vorab definiert werden (geht nur per DTD)

Kombinierende diakritische Zeichen

- Z.B. übergestelltes ᵘ ("COMBINING LATIN SMALL LETTER U" codepoint U +0367) = oͧ
- o͗ (oͧ) vs.  (Sonderzeichen im Zeichensatz mediaevum.ttf)
- Generelles Problem:
  - Ungewöhnliche Zeichen werden nur mit entsprechenden Zeichensätzen und entsprechender Software ordentlich angezeigt
  - Empfehlenswerte Schriften u.a. Arial Unicode MS, Junicode (http://junicode.sourceforge.net/), Code2000; MUFI (Medieaval Unicode Font Initiative, http://gandalf.aksis.uib.no/mufi/)
  - Weniger empfehlenswert: Mediaevum
- Z.T. gibt es mehrere Wege, ein und dasselbe Zeichen darzustellen:
  - Ä kann als A (U+0041) und Trema ¨ ('COMBINING DIAERESIS' = U+0308) oder als vorkombiniertes Ä ('LATIN CAPITAL LETTER A WITH DIAERESIS' = U +00C4) kodiert werden
  - Kann durch "Unicode Normalization Forms" gelöst werden (auch automatisiert)
  - http://www.unicode.org/reports/tr15/

Was tun, wenn Unicode nicht weiterhilft?

- A)
  - ohne weitere Deklaration die Private Use Areas (PUA) verwenden (U+E000-F8FF und die kompletten Unicode-Planes 16 u. 17)
  - den entprechenden Zeichensatz mitliefern
  - hoffen, dass schon alles klappen wird

- B)
  - TEI bietet mit den Elementen <char>, <glyph> und <g> eine flexible Methode zur Definition von Sonderzeichen und ggf. deren Umsetzung an
  - Character -> ein bestimmter „Buchstabe“ (z.B. ein A)
  - Glyph -> eine bestimmte Ausführung eines Buchstabens („langes s“, „rundes r“)
  - char und glyph können z.B. per XSLT auch in Sonderzeichen aus der PUA umgewandelt werden!

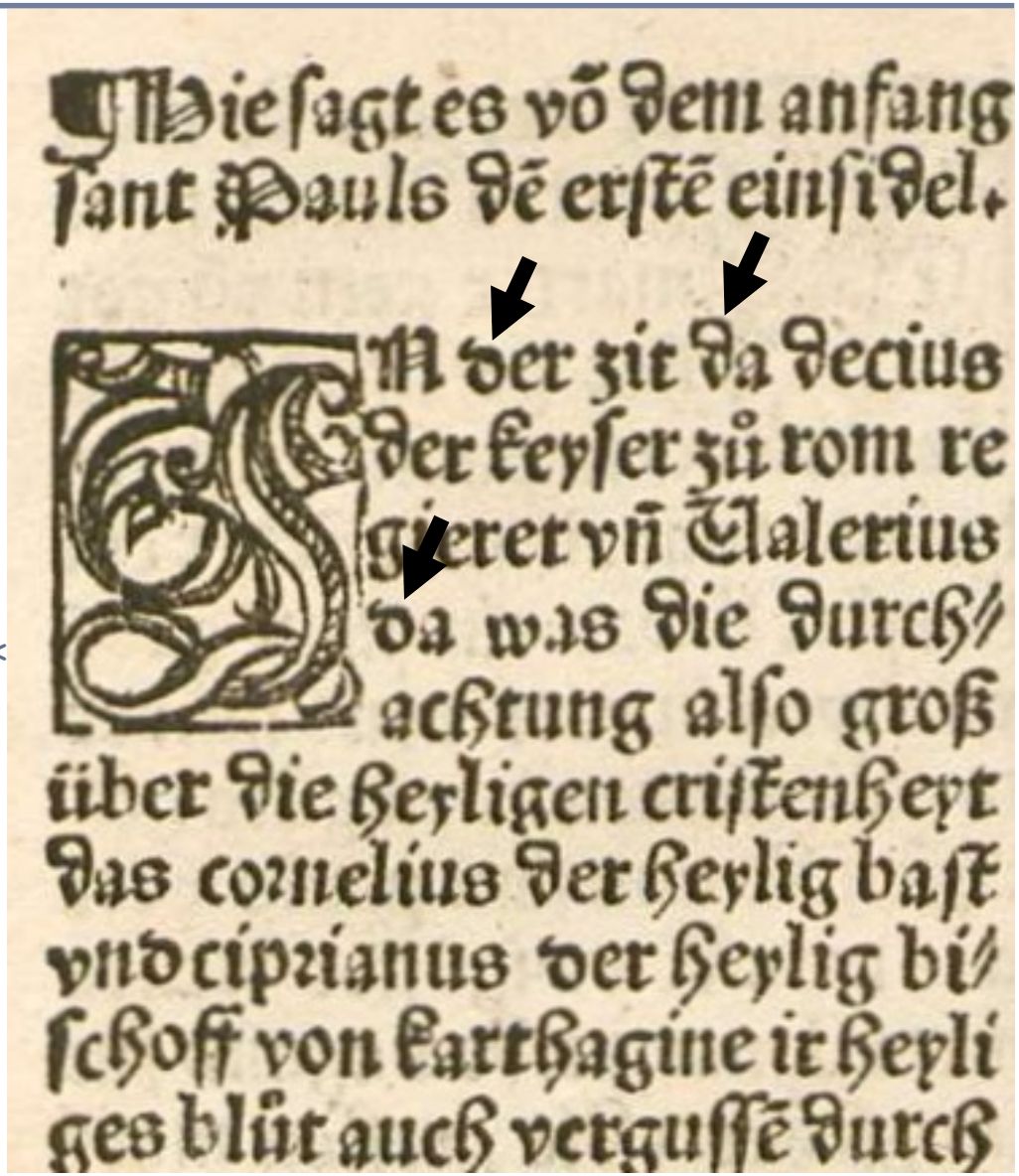
Das Element <charDecl>

- Teil von /TEI/teiHeader/encodingDesc
- Enthält <char>- und <glyph>-Elemente
- Darin u.a.:
  - <charName> bzw. <glyphName>
  - <charProp>
  - <desc>
  - <mapping>
  - <figure>

Ein Beispiel

```
<encodingDesc>
  …
  <charDecl>
    <glyph xml:id="variantd">
      <glyphName>VARIANT OF LATIN SMALL
                 LETTER D</glyphName>
      <desc>rundes kleines d</desc>
      <mapping type="standardized">d</mapping>
      <figure>
        <graphic url="variant-d.jpg"/>
      </figure>
      <note>nur am Wortanfang verwendet, selten<
    </glyph>
   </charDecl>
  …
</encodingDesc>
```

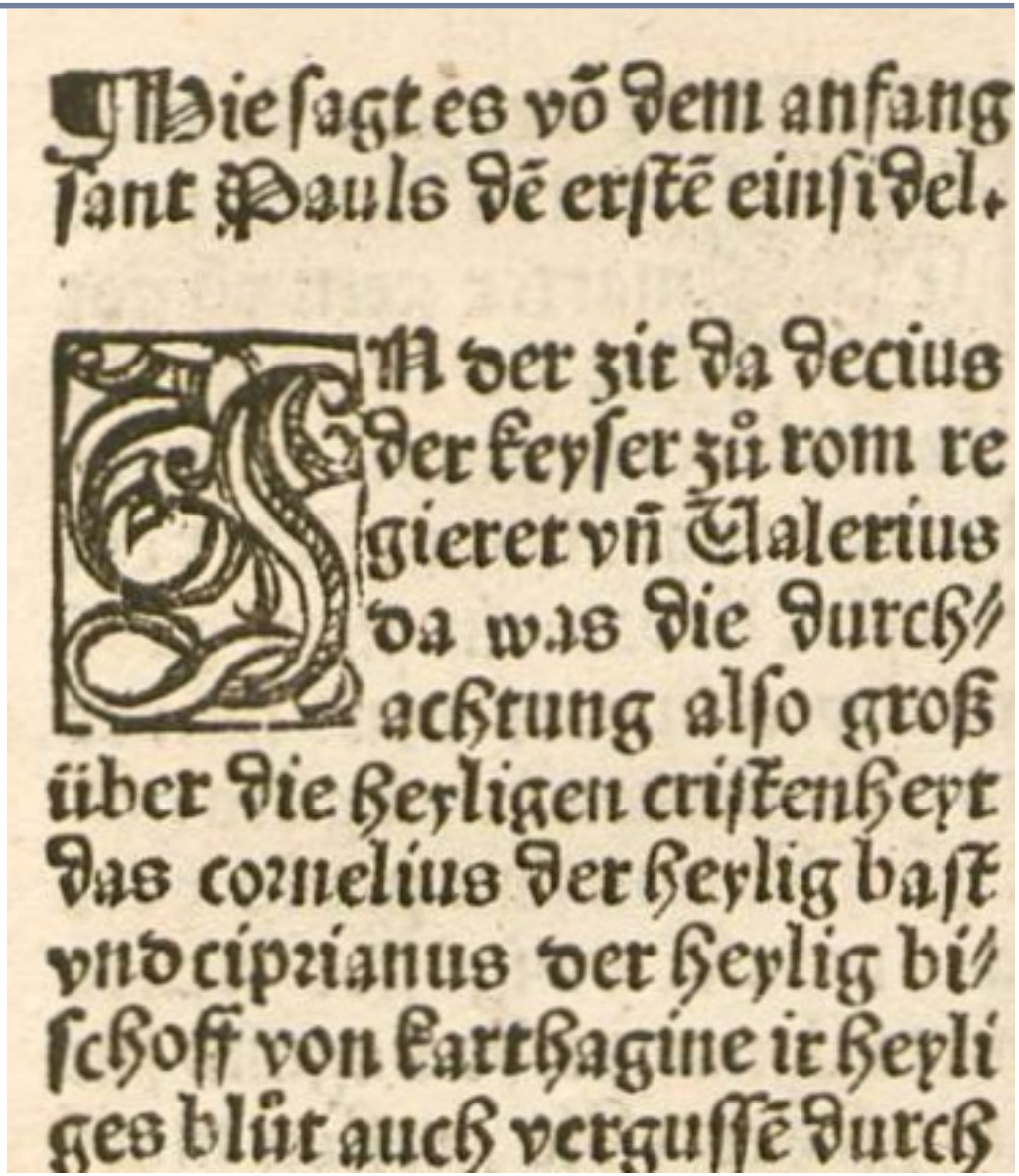
Vgl. gaiji.xml



Ein Beispiel

```
<TEI>
…
<text>
  <body>
    <p>
      …
      <lb n="3"/>Jn
       <g ref="#variantd">d</g>er
       zit da decius
      …
    </p>
  </body>
</text>
</TEI>
```

Beispiel: gaiji.xml
Beispieltransformation: gaiji2html.xsl



Übung

- Ermitteln Sie über die Seite http://www.fileformat.info/ den Codepoint für die Abbreviatur "per" (Unicode-Name: LATIN SMALL LETTER P WITH STROKE THROUGH DESCENDER)
- Finden Sie den entspechenden Codechart unter www.unicode.org/charts/
- Codieren Sie das Zeichen als Entität: &#xXXXX; und probieren sie aus, ob ihr PC das darstellen kann
- Ggf. Junicode installieren (http://junicode.sourceforge.net/)
- Codieren sie eine charDecl in der das Zeichen definiert wird und korrekt als Unicode-Zeichen aufgelöst wird (oder verändern sie das Beispiel „beispiellösung_gaiji.xml" dementsprechend)
- Wenden sie das Transformationsskript gaiji2html.xsl auf ihre Beispiellösung an
- Vgl. beispiellösung_gaiji.xml, gaiji2html.xsl