Unicode Myths

Mark Davis

Floating Myths

- Car companies are suppressing an engine that runs on water
- Neil Armstrong's walk on the moon was televised from a sound stage in Burbank, California
- Paying teachers by seniority—not merit—ensures a top-quality educational system

On to Unicode Myths

- Some of my personal favorites…

Conformance means supporting 95,000 Unicode characters

Informal restatement of conformance (FAQ):

- *It's OK to be ignorant about a character, but not plain wrong.*
- *Subsets are strictly up to you.*
- *Don't garble what you don't understand.*

Every Unicode code point represents a character

- Noncharacters:
  - FFFE, FFFF, 1FFFE,…
- Surrogate code points:
  - D800..DFFF
- Private Use code points: (maybe, maybe not…)
  - E000..F8FF,…
- Control/Format “characters”:
  - RLM, ZWNJ,…
- Unassigned code points

You can use any unassigned codepoint for internal use

- You are not conformant
- Software will break
- Eventually that hole will be filled with a different character

Use private use or noncharacters

“i18n” is not a geeky acronym

- This acronym has a long history, dating back to Capuchin monks
- Originally “iXIIXn” or “iXVIIIn”
- Purists prefer “i12n” (hexadecimal)
- Capuchin → Cappuccino
- Coincidence?

Unicode is missing characters for (Lithuanian/Yoruba/Slovak/...)

- Grapheme cluster ≠ code point
- “ch” *is* represented in Unicode: “c” + “h”
- “ẋ” *is* represented in Unicode: “x” + “◌̇”
- See Where is my character?

I don't need Unicode: 8 bits are enough

- Typographers say: 400-500 characters minimum for alphabetic languages
- Cross-platform data exchange is extremely difficult in the absence of Unicode:
  - There is no other common character set that is equally supported on Windows, Mac, & Unix

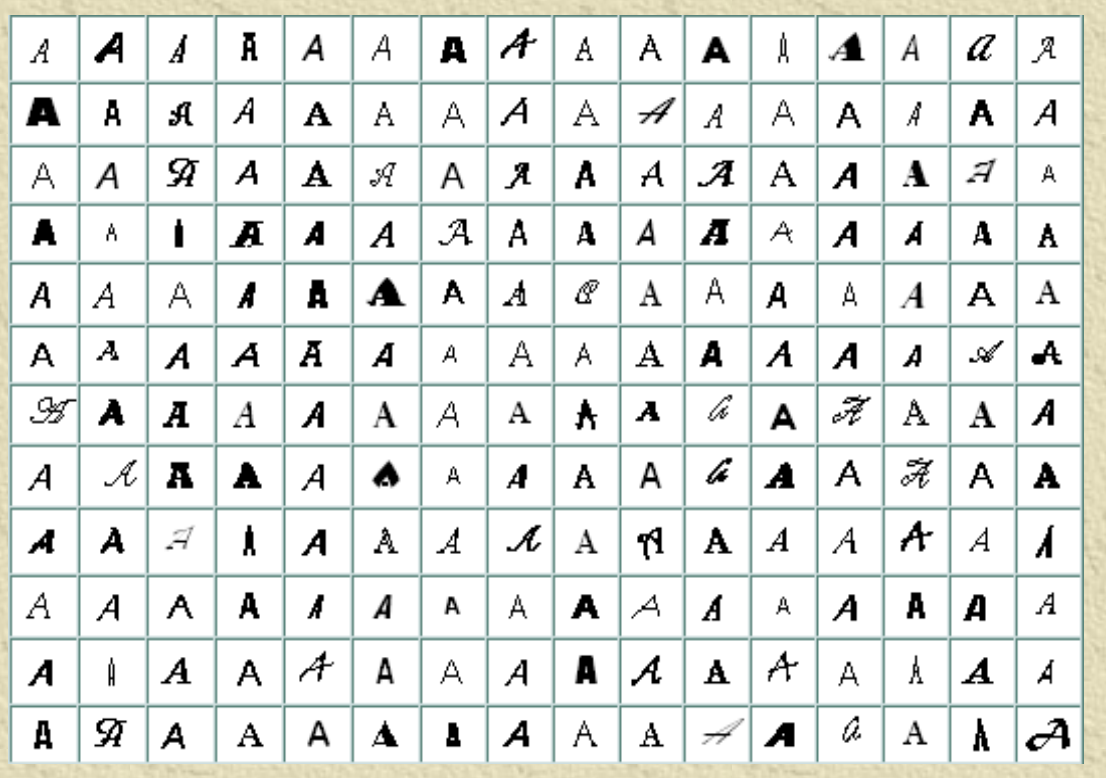
Unicode has character X in the wrong place; it sorts incorrectly

- Even for English, binary sort not right:

  “Z” < “a”

- Letters *cannot* be in the order of every language that uses a particular script
- For stability, Unicode Policy #1 never allows characters to be moved
  - See Unicode Policies, UTS #10: Collation

We need a glyph encoding standard…



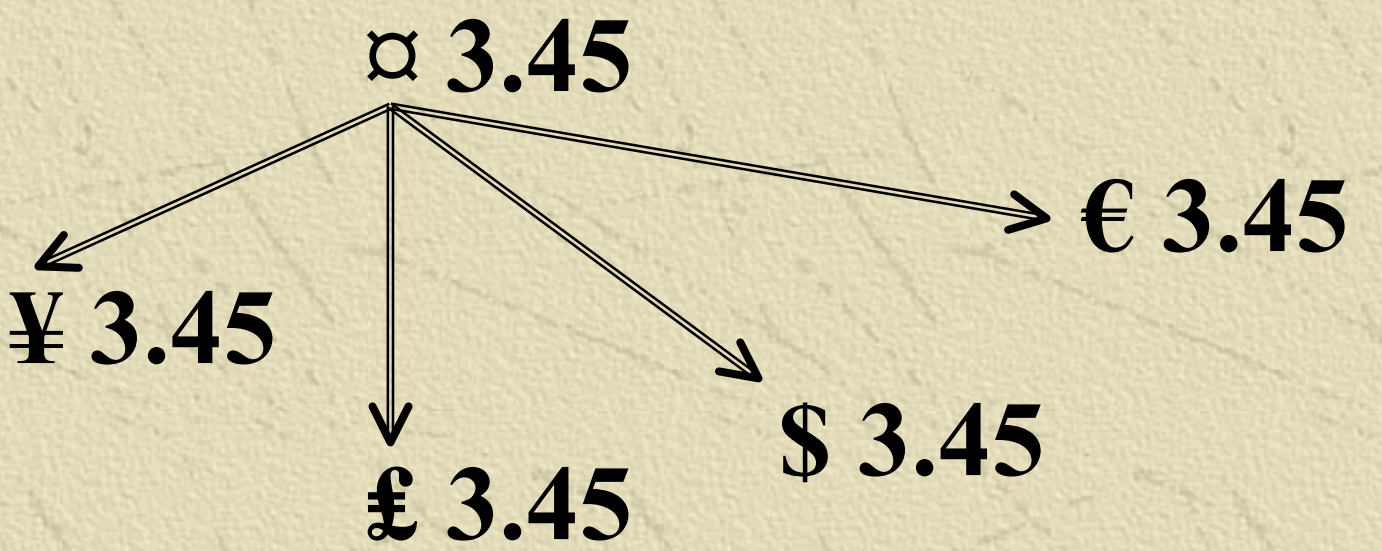
…So which glyphs?

To display a Unicode string, I need a complete Unicode font

- You need fonts that cover the characters in the string, plus *smart* programs:
- Systems can use hierarchical fonts
  - Latin → *Times*
  - Greek → *Athens, …*
- Programs, like IE, can use backup fonts
  - If a character is not in a font, they switch.

Unicode should have a “decimal point” character

- 3⌂456 → 3,456 or 3.456, depending on locale
- We considered radically “localizable shapes”
- The international currency symbol cured us



The Unicode Consortium doesn't care about Japanese*

- Almost 75% of Unicode are CJK
  - Includes all of the CJK characters from JIS X 0208, JIS X 0212, JIS X 0221, and JIS X 0213
- For CJK, the consortium follows the IRG
  - IRG members: China, Hong Kong (SAR), Macao (SAR), Singapore, Japan, South Korea, North Korea, Taiwan, Vietnam, US

* *Chinese, etc.*

Japanese* don't use Unicode

- People don't realize how prevalent Unicode is:
- *Clients:* Anyone using MS Office, Windows NT/2000/XP, MacOS, Java, or XML
- *Servers:* internally (to mix data from different languages without corruption);
  then serving up the data in desired code pages.

**Chinese, etc.*

Simplified and Traditional Chinese are unified

- Nope. These are explicitly *not* unified in Unicode.

Compatibility characters are all (good/bad: pick one)

- *isCompatibility<u>Character</u>*(C)
  ≡ C was encoded for compatibility
- *isCompatibility<u>Decomposable</u>*(C)
  ≡ *NFC*(C) ≠ *NFD*(C)

\+ *Positives:* Preservation of round-tripping with legacy encodings

– *Negatives:* Confusion, security risks

Unicode requires language tagging

- Unicode characters have identity *independent* of language
- Language information *can* improve communication: look at HTTP
- But few if any convincing scenarios where fine-grained, **plain-text** language tagging is required. (See §5.11)
- Use Markup

Fonts should put a Yen glyph at U+005C

- There are two codes for *yen sign:*

  U+00A5 (¥) Yen Sign

  U+FFE5 (￥) Fullwidth Yen Sign

- U+005C \ Reverse Solidus is a backslash; a *yen sign* glyph there is compatibility hack

Normalized text (NFC) does not contain combining marks

- Wrong – "Ẋ" is <0078, 0323>
- For stability, new composite characters (post 3.0) are *decomposed* in NFC
  - See UAX #15: Normalization
- Note: combining marks are required
  - Representation of many languages: Arabic, Thai, …
  - Generative use in linguistics, mathematics, …

Liberal use of the word MARK in character names is my doing

- Just a coïncidence…

Unicode was invented by printer manufacturers

- News to me…and I was there!
- Microsoft and IBM are not usually thought of as "printer companies"
- From the beginning, the goal was for both internal process code *and* interchange
- Items that *look* like printer-ROM hacks (e.g., Zapf dingbats), were pushed by software companies like WordPerfect—not HP, Apple, or Xerox

Unicode has KELVIN sign, but is missing many other units

- Unicode does *not* distinguish letters by usage
  - *Kelvin*, *Ohm*, etc. are discouraged; encoded purely for compatibility
- No difference in code points:
  - “g”: go vs. 12g
  - “Å”: Århus vs. 15Å
  - “U”: Underwood vs. UTF-8
  - “e”: exception vs. $x + e^3$

Original design (16-bits) was never possible

- Goal *was:*
  - commercially significant characters of the world
- Strategy *was:*
  - Mechanisms for uncommon CJK (e.g. IDS/variation selection)
  - Few composites (no Hangul Syl., Arabic ligatures, &c.)
  - PUA for archaic scripts, uncommon symbols, etc.
- Of course, the goals changed dramatically!

Case mappings are 1-1

- One-to-many: **ß → SS**
- Contextual: **…Σ ↔ …ς**
  …ΣT… ↔ …στ…
- Locale-sensitive: **I ↔ ı**
  İ ↔ i
  - See UTR #21: Case Mapping, Charts

Korea wants 41 new variant forms of the “Turtle” character

- Actually, this one is true; and in addition to the current 7 variants of:

U+9F9C 龜 “turtle”

- This is one of the reasons for the addition of the *Variation Selectors* in Unicode 3.2

UTF-x is better than UTF-y

- ***All are Unicode!***
- The best one depends on the environment:
  - UTF-8: most legacy-compatible, least simple
  - UTF-16: good compromise on ease, storage
  - UTF-32 easiest to process, most storage
    - Note: some of the “ease” of UTF-32 is illusory
    - Most code needs *strings,* not code points

You don’t need to worry about supplementary characters

- Depends on who you are!
- Supplementary characters (above FFFF) have arrived — Unicode 3.1
- Some are required by East Asian governments

Unicode will run out of space

- 974,530 code points (– PUA/NCCP/Surr.)
- 95,212 assigned in 15 years
- If it *were* linear
  - In 2140 AD, we would run dry
- *But it isn't…*

See http://www.unicode.org/roadmaps/

Unicode doesn't support Indic Half-Forms

- Half-forms ***are*** represented, with

  Consonant + Halant

- Example:

  U+0924 DEVANAGARI LETTER TA +
  U+094D DEVANAGARI SIGN VIRAMA
  (= halant)

Sarasvati is really Rasa Tavis

- ***False:*** here at her home in Delhi:

You will have to rewrite all your code for surrogates

- Important:
  - ***Surrogates don't overlap!***
  - ***Most code not sensitive to surrogates***
  - ***Good code accounts for strings, not just code points***
- The String datatype doesn't have to change
- Based on experience, well written code should not need many changes

Unicode is overly complex

"Everything should be as simple as possible, but no simpler." – Albert Einstein*

- Bidirectional scripts, combining marks, large character sets, collation, …
  - *Human writing systems **are** simply complex*
- Byte order, encoding forms, compatibility characters, Yen vs. backslash, …
  - *Compatibility was **required** with legacy encodings / systems*

Unicode is overly complex (II)

- There are areas, in retrospect, that could have been better,
- But in any event, most people are insulated from the complexities by code libraries
  - *Java, Windows, ICU, Rosette, ...*

Unicode is pronounced “you knee code”

- No, instead, rhymes with “you nick code”
  - Unconnected with “nick” (meaning “steal”)
  - Or “unique”
  - Or “Unix”, “Eunuchs”
- In IPA: ˈjuːnɪkoʊd
- For Americans: Ūn̆icōdĕ

For many more myths,

http://www.unicode.org/faq