How Unicode Came to "Dominate the World"

Lee Collins

18 September 2014

Overview

- Original design of Unicode
- Compromises
  - Technical
    - To correct flaws
  - Political
    - To buy votes
- Dominates the world
  - But is it still “Unicode”

Why Unicode

- Mid-late 1980s growth of internationalization
  - Spread of personal computer
- Frustration with existing character encodings
  - ISO / IEC 2022-based (ISO 8895, Xerox)
  - Font-based (Mac)
  - Code pages (Windows)

Existing Encodings

- No single standard
  - Different solutions based on single language
  - Complex multibyte encodings
    - ISO 2022, Shift JIS, etc.
- Multilinguality virtually impossible
- Barrier to design of internationalization libraries

Assumptions

- Encoding is foundation of layered model
  - Simple, stable base for complex processing
- Characters have only ideal shape
  - Final shape realized in glyphs
    - Font, family, weight, context
- Character properties
  - Directionality
  - Interaction with surrounding characters
- Non-properties
  - Language, order in collation sequence, etc.
  - Depend on context

Unicode Design

- Single character set
  - Sufficient for living languages
- Simple encoding model
  - “Begin at zero and add next character” — Peter Fenwick of BSI at Xerox 1987
  - No character set shift sequences or mechanisms
    - Font, code page or ISO 2022 style
  - Fixed width of 16 bits
- Encode only atomic elements
  - Assume sophisticated rendering technology
    - a +˘ + ´ = ắ
    - र ् क ् ष = क्ष

Early Strategy

- Unicode as pivot code
  - Interchange between existing encodings
- Focus on particular OSs
  - Xerox, Mac, NeXTSTEP, Windows
- Build libraries to bootstrap adoption
  - Character set converters
  - Drivers to read and write Unicode files
  - Fonts and rendering
    - TrueType, QuickDraw GX

Plain Text Debate

- Test to determine atomicity of related characters
- No agreed criteria for application
  - Han requires multiple fonts to achieve same effect as Greek and Coptic
- Used to bypass original model
- Alternative is variation selectors

Before Standardization

- Character set standard bodies
  - Dominated by large, established companies and governments
    - IBM, DEC, HP, Honeywell
    - JIS, GB
  - Very formal process
- Unicode informal meetings
  - Word of mouth
    - Engineers from Xerox, Apple, Metaphor, NeXT, RLG, Sun
  - “No stinking process”
    - Uncontrolled debate and agenda
    - Legal issues?
  - Let the standards guys come to us

ANSI X3L2 September 1988

- Unicode to ISO
  - You guys got it all wrong, start over with Unicode
- Critique of ISO 10646
  - No Han-Unification
  - No character composition model
  - Bidi-model
  - Complexity of shifting between planes
- Began a long period of heated debate
  - X3L2 eventually supported Unicode
    - Members used global influence
- Unicode now involved in ISO 10646 standardization process

Key Disagreements

- Han Unification
  - 10646
    - Independent sets for China, Japan, Korea
    - Left out Hong Kong and Taiwan
- Composition
  - 10646 explicitly rejected combining marks
- Outcomes seriously affected Unicode

Han Unification

- Corollary of the Unicode character model
  - Ideal shape
  - Basic properties do not include
    - Inherent language
    - Position in collation sequence
    - Reading, etc.
- Saving code points was nice side effect
  - Esp. considering extensions

Reaction to Unification

- China
  - Parallel unification effort under Zhang Zhoucai
- Japan
  - Very controversial
    - Why are gaijin telling us how to encode our characters?
    - Some supporters, esp. librarians (NACSIS)
- Korea
  - Ambivalent
- Taiwan
  - Mostly supportive
    - Already encoding Chinese and Japanese forms in own standard
- Vietnam
  - Yes, but add Chữ Nôm

Closure on Unification

- Role of IBM
  - Cooperation between Unicode and China
  - Validation by Prof. Nakajima
  - Helped bring Japan to table
- Formation of CJK-JRG
  - Japan a reluctant participant

Outcome of Unification

- Unification model
- Round-trip mapping rule
  - Many exceptions to unification model
    - Need to look at source set to understand
      - 說　kIRG_TSource　= T1-6B29
      - 説　kIRG_TSource　= T3-4966
  - Spawned creation of ad-hoc “standards”
- On-going standardization via IRG

Composition

- Cultural reaction
  - Europe: ä is a letter in my alphabet
    - Needs single code point
  - India: क्ष is an akṣara in the varṇamālā
    - I know it's composed
  - East Asia: Any blob surrounded by space is character
- Technical objections
  - Too hard too implement
  - Performance and storage

Composition Outcome?

- Both forms allowed
  - For some scripts
- Cultural reactions dominated
  - Roman, Greek, Cyrillic
  - Korean, Vietnamese
  - Tibetan
    - Chinese wanted full composition
- Additional normalization requirement

Unicode - ISO 10646 Merger

- Cost to Unicode
  - Loosening of principles
  - More complex process
- Critical for success
  - Authority of ISO standards
    - adopted by governments and businesses

Extending Code Space

- Originally fixed-width, 16-bit
  - Feasible based on original model
- Less viable after compromises
  - 11K Korean hangul
  - IRG started allowing large # of variants
- Ancient scripts

Surrogates

- 1996
- Solved the encoding space problem
- Simple, fixed-width?
  - No
    - Need to test value and get next surrogate
  - Yes
    - Surrogate pairs = 2 fixed-width codes points
    - Unique: no scanning to determine value
- Works, but complicates the original layering model

UTF-8

- Pike and Thompson 1992
- Solution for existing systems and languages
- 8-bit safe
  - esp. null and /
- No endian issues
- Multiple width, but self-synchronizing
    - Easy to determine position in string
- Critical to adoption and spread of Unicode on the Web
  - Most Unicode data on web is UTF-8

Java

- 1995
- 16-bit Unicode characters
- Unicode-based string class
- Main language for web services

ICU

- IBMs portable open-source i18n library
  - Came out of Pink and Taligent
- Ported to Java
  - Base for Java's i18n support
- Facilitated adoption of Unicode
  - Hides dirty details
  - Rich set of features
- Now basis for i18n in many places
  - OSs, languages, web services, etc.

CLDR

- Unicode-based locale data
  - Date, time, number formats…
  - Raw data for ICU
- Open source
  - Contributors encouraged

NeXTSTEP

- 1988 - 1996
- Rich set of Unicode classes
  - Factor in Apple’s choice of NeXT over BeOS
- Morphed into Mac OSX and iOS

Windows NT

- 1993
- First major OS with Unicode support
  - Internal encoding
- Continued in later releases
  - XP, Vista, 8, etc.

Growth of Web

- 1990s
- Explosive demand for local language support
- Unicode support in search engines, browsers
  - Webkit
- W3C standards
  - Unicode recommended as default for XML and HTML
- Unicode now 80% of Web data?

Internet Services

- Early 2000s
- Social media
  - Facebook, Twitter
- Video sharing and streaming
  - YouTube, Netflix
- Music
  - iTunes
- Finance
  - Paypal

Smart Phones

- 2007 iPhone revolution
- Mobile computing power
  - In hands of people everywhere
    - Esp. non-PC users
- Everyone using iOS, Android, WP is using Unicode
  - Texting, tweeting, browsing, connecting with friends…

Emoji

- Most exciting Unicode event in recent years
  - Universally popular
    - Driven by smart phones
  - PR for Unicode
    - And some controversy
  - Outside of BMP
    - U+1F5FB…
- Pressure to support full Unicode (surrogates)
  - SMP: No longer rare characters

Current Challenges

- Competing standards
- Implementation issues

Competitors

- China
  - Government pushing local standards
    - GB 18030, GB/T20524-2006, etc.
      - Internally can use Unicode
    - Local OSs: Kylin, Red Star
- Japan
  - TRON
    - OS for everything:
      - Appliances, feature phones, etc.
    - Limited to Japan

Implementation Issues

- Clumsy support in some popular programming languages
  - Not default
  - Surrogate support
- Old mail SW
  - Esp. Japan
- Incorrectly tagged web content
- Lots of minor variants
  - 魑, 魑, 魛, 魛…
  - Need normalization for any search

Outlook

- China's massive web presence could change things
  - If local systems and services move to GB
- Implementation issues
  - Solvable
    - Esp. given pressure to support Emoji
  - New languages like Apple's Swift provide hope

Is it still Unicode?

- Universal
- Uniform
- Unique

Universal

- Does it fill needs of world languages?
- Yes
  - Can’t claim all, but certainly most
  - New scripts and characters for languages being added
  - Living and dead

Uniform

- Fixed-width for efficient access?
- Yes and no
  - UTF-32
    - Original meaning of fixed-width
  - UTF-16
    - A pain to iterate character by character
  - UTF-8
    - Not fixed-width, but self synching
- All are improvements over Unicode's predecessors

Unique

- Single interpretation of bit sequence
- Yes
  - Once you identify the form

Conclusion

- Wildly successful
- Has changed greatly
  - Big, complex, messy to implement
  - Library and language support critical
- Challenges remain
- Unlikely that growth of adoption will change