Integrating the development of encoding, font, and keyboard

Norbert Lindenberg

Problem 1: Mongolian

- 1999: Mongolian in Unicode 3.0
- 2018: *The Mongolian script: What’s going on?*

Problem 2: Myanmar

- 1999: Unicode 3.0 – Burmese only
- 2002: Myazedi; 2006: Zawgyi – hacked Unicode
- 2008: OpenType (mymr), HarfBuzz
- 2008: Unicode 5.1 – minority languages
- 2013: OpenType (mym2), Windows 8.1
- 2018: *Fixing Burmese: Dealing with Zawgyi*

Problem 3: Khmer

- 1999: Unicode 3.0
- 2002/2007: OpenType
- 2018: What’s a valid Khmer syllable?
  - Unicode: B {R | C} {S {R}}* {{Z} V } {O} {S}
  - OpenType: Cons + {COENG + (Cons | IndV)} + [PreV | BlwV] + [RegShift] + [AbvV] + {AbvS} + [PstV] + [PstS]

Problem 3: Khmer

- Khmer syllable according to Unicode: (Cons | IndV) [R | RegShift] {COENG (Cons | IndV) [R]} [[Z] (PreV | BlwV | AbvV | PstV)] [AbvS | PstS] [COENG (Cons | IndV)]
- Khmer syllable according to OpenType: Cons {COENG (Cons | IndV)} [PreV | BlwV] [RegShift] [AbvV] {AbvS} [PstV] [PstS]

Script encoding process

- Various people write character proposals
- UTC and WG2 accept or reject proposals
- UTC and editorial committee produce Standard & data

- Various people try to implement fonts, keyboards, supporting libraries
- Eventually Microsoft defines OpenType rules
- More people create fonts based on OpenType

Unicode proposals

- Show characters to be encoded
- Come with font that’s just good enough to create proposal
- Have some information about usage, but rarely enough to create functional font
- Do not come with keyboard
- Have some data, but not enough for libraries

Other standards processes

- IETF: Rough consensus and running code
- W3C: Implementation experience is required
- Ecma TC39: Two compatible implementations passing acceptance tests

Making a complex script work

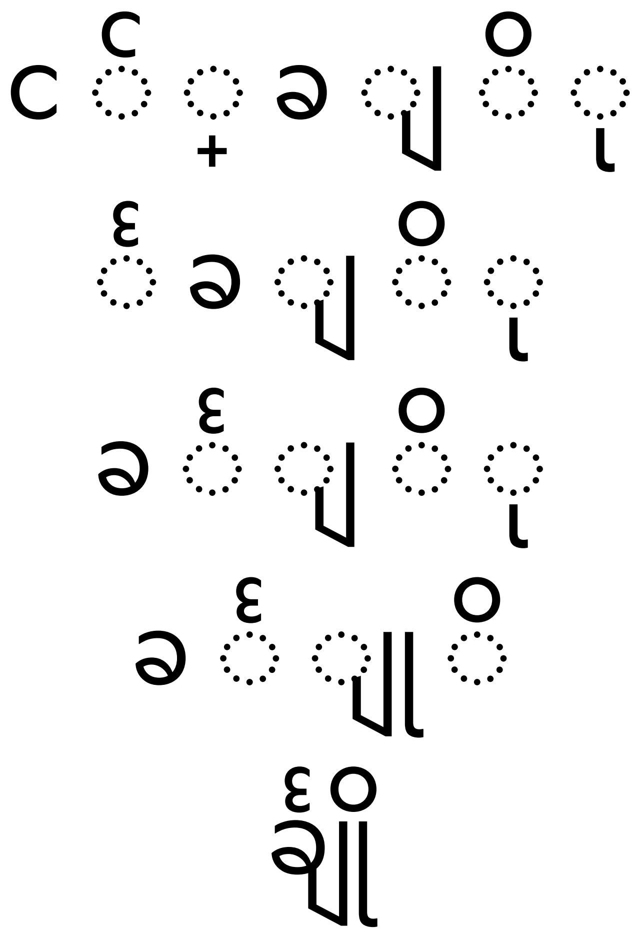
- Font; font technology
- Keyboard; keyboard framework
- Rules for line breaking; maybe with dictionary
- Definition of units for selection, deletion
- Comparison rules for search
- Comparison rules for sorting

Making a complex script work

- Font; font technology
- Keyboard; keyboard framework
- Rules for line breaking; maybe with dictionary
- Definition of units for selection, deletion
- Comparison rules for search
- Comparison rules for sorting

Fonts

Why is it complicated?



What’s needed

- Complete specification of structure of valid clusters
- Complete specification of required shaping
- Description of common presentation choices
- Compilation of these specifications into fonts

Khmer clusters?

(Cons | IndV | GB) [RobatGroup | ([CoengGroup] [TailGroup])]

RobatGroup: R ([PreV | BlwV | PstV] | PstS)

CoengGroup: (BlwC [BlwC | PstC | PreC]) | (PstC [PstC | PreC]) | PreC

TailGroup: [RegShift] [[[PreV | BlwV | AbvV] [AbvS] [PstV] [AbvS] [PstS]] | [PstV [AbvS] (BlwC | PstC)]]

Myanmar required shaping

```
sub nga asat virama by kinzi

sub kinzi @bases by $2 $1

sub yaMedial u' by u.post ignoring
@above_base
```

Implementation

- Apple Advanced Typography – iOS and macOS; maybe soon in HarfBuzz
- Graphite – Firefox
- OpenType once Universal Shaping Engine works for the script – everywhere
- Compile validation and shaping rules to rule languages for these technologies

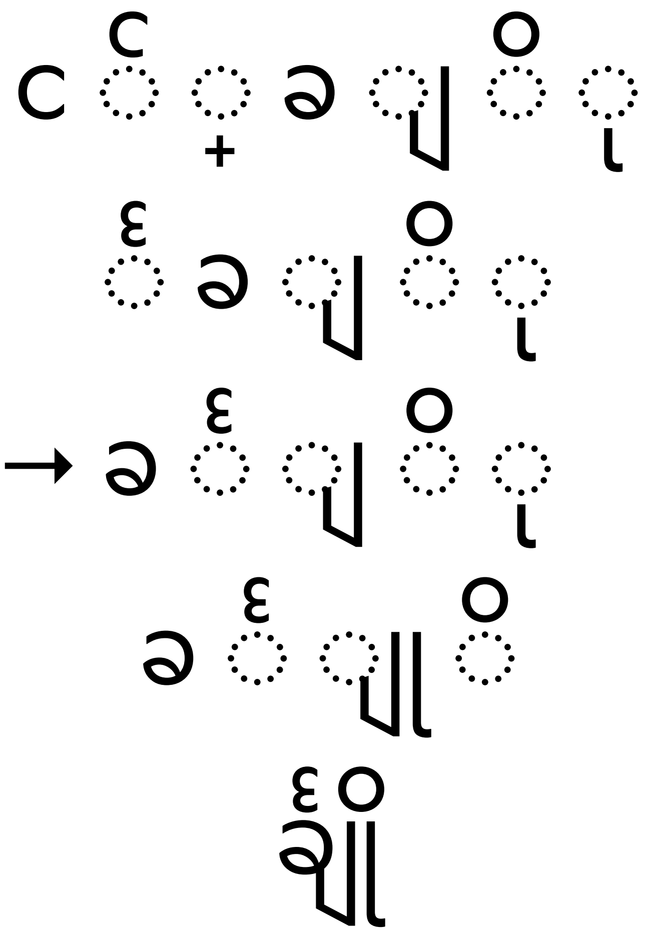
Keyboards

Why is it complicated?

Unicode →

User level components →

Display →



Issues

- Unicode characters are not always the text units that users perceive
- Users don’t always type characters in the order Unicode and OpenType expect
- Users know nothing about Unicode normalization

Opportunities

- On mobile devices, keyboards show user what they can type
  - No need to map to Latin
- Keyboards can maintain context, reorder and fix input

Formal description

- Latest version of Locale Data Markup Language for keyboards includes reordering, support for multi-character units
- Behavior can be specified separately from layout

Keyman

- Cross-platform keyboard framework for Windows, Android, iOS, macOS, web
- Started by Tavultesoft, acquired by SIL International
- Roadmap targets LDML support for version 12, early 2019

Testing

- Provide users with fonts and keyboards
    - Android: font packaged into app with editor
    - Other platforms: Installable for use in any app
- Watch how well they meet their needs

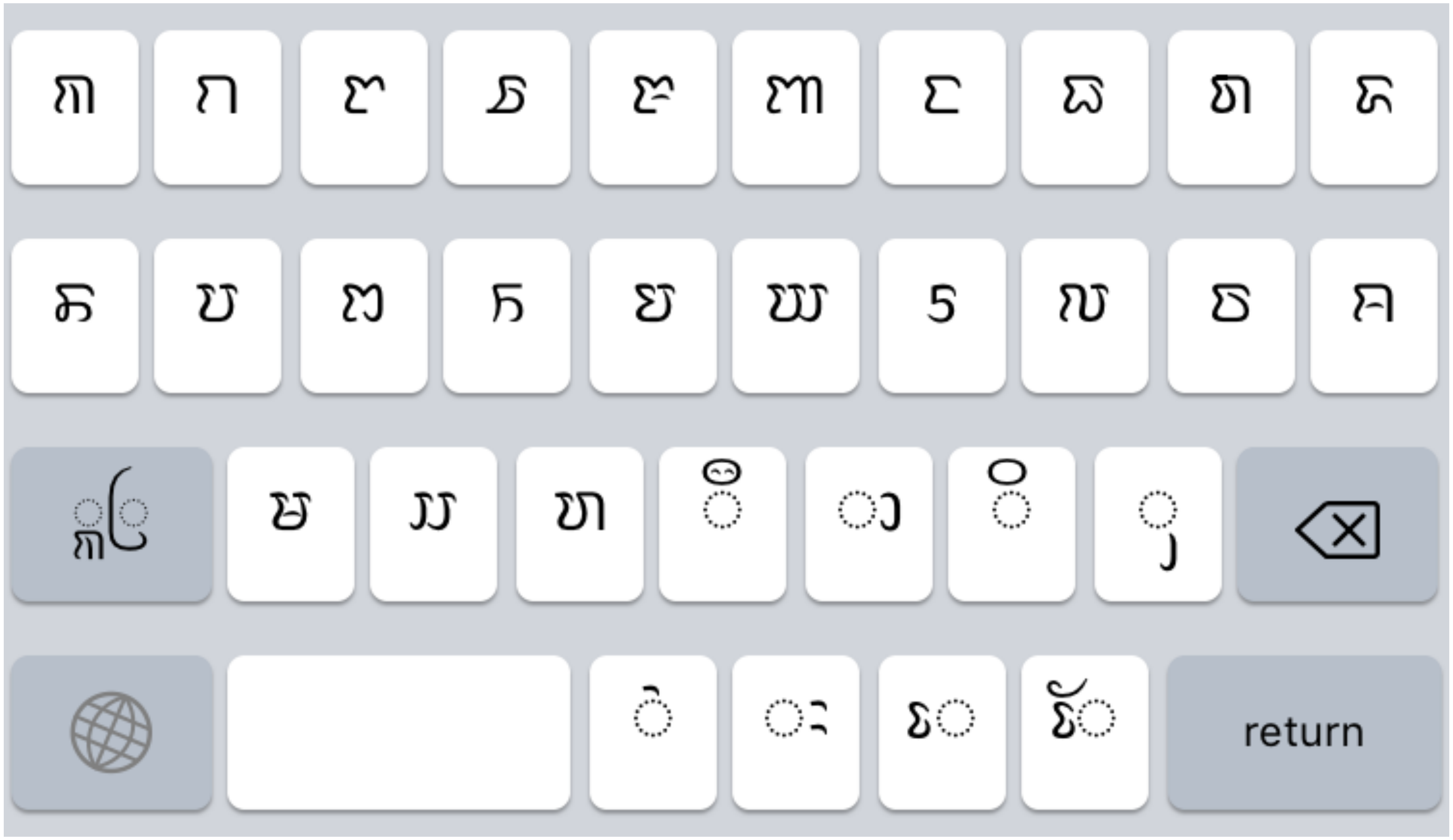
Compatibility

- Encoding may need to change in response to test results
- Might use private use area during testing to avoid compatibility issues
    - But: Shaping fails on Windows with private use area fonts

Example: Kawi

- Brahmic script from Java, used 8th – 15th century in much of today’s Indonesia and parts of Philippines
- Preliminary Unicode proposal from 2012
- Implemented here in PUA, U+F1DB0–F1DFF, on iOS using AAT
- Font design by Aditya Bayu Perdana

[illegible]


5
return

Discuss.

Fonts used

- Sanomat Burmese and Khmer

  Sanomat Burmese and Khmer extensions produced by The Fontpad Ltd. Project management, technical support and feedback by Ben Mitchell. Design by Mark Frömberg and Natalie Rauch. Khmer consultancy by Sovichet Tep. Engineering by Norbert Lindenberg. Original Latin design by Vincent Chan and Christian Schwartz at Commercial Type. Thai extension by Smich Smanloh at Cadson Demak.

- Kawi demo

  Design by Aditya Bayu Perdana. Engineering by Norbert Lindenberg.

- Myriad Pro

  Design by Robert Slimbach and Carol Twombly at Adobe Systems Inc.