

Creating fonts for Brahmic scripts with OpenType and Apple Advanced Typography

Muthu Nedumaran & Norbert Lindenberg

Agenda

- What are Brahmic scripts?
- Creating glyphs
- Unicode and fonts
- OpenType shaping
- Apple Advanced Typography shaping

What are Brahmic scripts?

- Scripts derived from Brahmi (~250 BCE)
- India: देवनागरी, বাংলা লিপি, தமிழ், ...
- Around India: ཨིཾཨྱཾ, འོད་ཡིག་, ...
- SE Asia: ไทย, 𑄆𑄓𑄗𑄓, 𑀅𑀲𑀯𑀭𑀮𑀶𑀢𑀺, ...
- East Asia: Siddham

Brahmic scripts

Note: The following pages use a font that exhibits properties common to many Brahmic scripts but doesn't fully represent any of them.




Brahmic scripts

- Abugida: consonants have inherent vowels
 - ◻ *sa*, ◻ *ta*, ◻ *ra*
- Inherent vowel can be overridden with dependent vowel mark (matra)
 - ◻ *sa*, ◻̂ *si*, ◻̣ *su*, ◻̥ *se*, ◻̦ *so*
- Independent vowels also exist

Brahmic scripts

- Inherent vowel can be eliminated, e.g. *sta*
 - Visible vowel killer (virama): □ζ◊
 - Half-form: ◻◊
 - Conjunct: ◻
 - Subjoined consonant: ◻
 - Postfix consonant, dropped consonant, reph, ...

Brahmic scripts

- Numerous additional marks
 - Anusvara, chandrabindu, nukta, visarga
 - Special medial consonants ya, wa, ra, ...
- Occur above, below, or after base
- Medial ra wraps around in some scripts: 
- Overall: Far more complex than Latin script

Creating glyphs

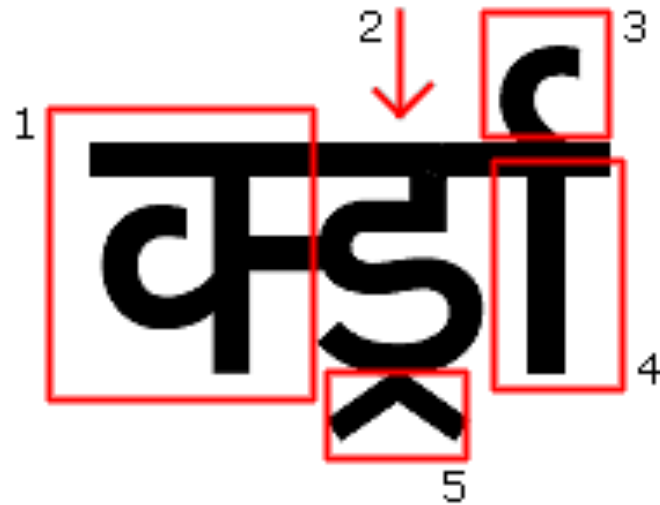
Creating glyphs

- Get a good understanding of the script
 - History, forms, proportions
- Language variants
 - Same Unicode character, different form
- Trends and typographic needs
 - Contrast and modulation
 - Weights and styles

Creating glyphs: Best practices

- Don't create complete clusters
 - Work on characters and marks
 - Use them as components
 - Use substitution and positioning to form clusters
- Create pre-composed forms for complex shapes

Example



source:

<https://www.microsoft.com/typography/OpenTypeDev/devanagari/intro.htm>

Spacing

- Many scripts have head lines rather than base line:
 - eg: हिन्दी
- Base glyphs may have layers of marks on top and bottom
 - Marks can be vowel signs, subjoined consonants or diacritics
- Metrics used for Latin may not work for Brahmic scripts


Example: base + marks



source:

https://cdac.in/index.aspx?id=mlc_gist_font

Fonts for UI

- Generally low-contrast
 - Some scripts come from calligraphic tradition
 - Low-contrast can be a challenge
- Line heights generally fixed for Latin
 - May need to shrink above and below marks
 - May need to split stacked conjuncts 
 - May need to shift the baseline for Brahmic

Example: Hindi & English on the same line

.....यह पाठ हिंदी में है..... this text is in Hindi.....

Including Latin glyphs: Pros

- Some Latin is often mixed with non-Latin text
 - Good idea to include matching latin glyphs
 - English alone will suffice most of S.E.A.
 - Depending on the script, the x-height may not match the general height of Brahmic base letters

Including Latin glyphs:

Cons

- Including a complete Latin glyph repertoire can be laborious
- Limiting to English alone can leave other languages incomplete – like Vietnamese
- If a bigger glyph set is needed, it may be better to find a good Latin font that matches the Brahmic outlines
- Focus on what you do best!

Examples

- Vowel signs & diacritics
 - ໂ, ໄ, ອ, ັ, ື, ູ, ຸ, ົ, ື
- Subjoined consonants
 - ັ, ື, ຸ
- Marks attaching to subjoin consonants
 - ັ, ື

Unicode and fonts

Unicode encoding

- Unicode encodes *characters*
- Font provides *glyphs*
 - Various alternate glyphs for same character
 - Ligatures combining several characters into one glyph
- Rendering system and font interact to produce visual representation

Unicode encoding of Brahmic scripts

- Characters usually in phonetic order
 - Representation of *se* is *se*
 - Exceptions: Thai, Lao in visual order
- Conjuncts usually encoded with virama
 - Encoding of *sta* is *st* *virama*
 - Exception: Tibetan encodes subjoined consonants

Shaping

101E	1039	1010	103C	1031	102C
□	⚡	⬡	⌋	Ꞑ	ꞑ
ꞑꞐꞑ					

Shaping issues

- Decomposition
- Reordering
- Conjunct formation
- Other contextual forms
- Positioning

Shaping support

- AAT: do it yourself, low level, flexible
- Graphite: do it yourself, high level, flexible
- OpenType
 - script-specific shaping engines
 - decomposition and reordering handled (mostly) by engine

What's in a font?

- Font file contains many tables
- Outlines: glyf (TrueType), CFF (PostScript)
- Character-to-glyph mapping: cmap
- Shaping tables
 - OpenType: GSUB/GPOS/GDEF
 - AAT: morx/kerx/ankr
 - Graphite: Silf/Glat/Gloc

OpenType Shaping

Shaping tables

- GDEF: Classification of glyphs into bases, marks, ligatures, others; other sets
 - Not based on Unicode properties
- GSUB: substitutions – 1:1, 1:many, many:1, contextual
- GPOS: positioning – base to mark, mark to mark, kerning, cursive attachment

Shaping context

- Split text into font/style runs, script/language runs, clusters
- Map characters to glyphs with cmap table
- Let script shaping engine do its work, interpret GDEF/GSUB/GPOS tables
- Rasterize glyphs and render them in locations calculated by shaping

Shaping context

Text	Englishதமிழ்ஂூ			
Font 1	English			
Font 2		தமிழ்ஂூ		
Script run		தமிழ்		ஂூ
Cluster		ு	மி	ஂ்

Shaping engines

- OpenType renderer has 20 shaping engines
 - 13 for Brahmic scripts
- Each handles one or more scripts
- Each defines a set of shaping features
- Each provides some automatic behavior and applies the font's features

Shaping feature

- Originally an optional font feature, such as SMALL CAPS
- Set of change rules (“lookups”) to be applied to glyph sequence
- Selected by script and language, rarely by user
- Semantic names help organize rules
 - locl – language-specific forms
 - half – combine consonant and virama to half-form
- But mostly define *when* rules will be applied

Shaping phases

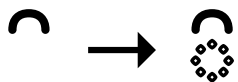
- Shaping engine processes cluster in several phases
- Some phases are automatic, based on Unicode data
- Some process rules for specific features provided by font
- Some do both

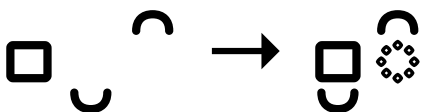
Shaping phases

- Cluster validation
- (De)composition, localized forms – GSUB
- Reordering – GSUB
- Basic shaping – GSUB
- Topographical features – GSUB
- Presentation forms – GSUB
- Positioning – GPOS

Cluster validation

- Enforces that marks follow bases in correct order
- Inserts dotted circle before invalid marks
- Automatic, based on Unicode character data

• 

• 

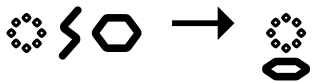

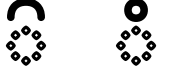

(De)composition, localized forms

- Decomposition partially automatic based on Unicode data, e.g., $\text{c} \circ \circ \text{d} \rightarrow \text{c} \circ \circ \quad \circ \circ \text{d}$
- (De)composition feature ccmp: اَلْجَزْ
 - sub sa-bali.conj by uMark-bali pa-bali.conj;
- Language-specific forms feature locl
 - sub nine-deva by nine-deva.nepali;

Reordering

- Mostly automatic based on Unicode data
 - Pre-base vowels: $\square c \rightarrow c \square$
 - Medial ra: $\square \text{U} \rightarrow \text{U} \square$
- Some glyphs identified by rphf, pref features
- May come before or after (de)composition, or after basic shaping, depending on engine




Basic shaping

- Features: abvf, blwf, pstf, and others
- Form conjuncts, e.g.  → 
 - sub virama ta by ta.conj;
- Combine side-by-side marks:  → 
 - lookupflag IgnoreBaseGlyphs UseMarkFilteringSet @above_base;
sub iMark anusvara by iMark_anusvara;
- Replace non-spacing marks with spacing glyphs

Topographical shaping

- Create initial, medial, final, standalone forms of clusters
- Only supported for Bengali and (theoretically) in USE



Presentation forms

- Bring glyphs into final forms
- Contextual alternates, separated by position:
pres, abvs, blws, psts, calt:  +  → 
- sub ra.below' @consonants @conjuncts by ra.below.low;
- Entire run visible for Myanmar and USE;
implementation dependent for other engines

Positioning

- Move glyphs into the right positions relative to each other, using anchors: mark, mkmk



- Adjust spacing: kern, dist:  → 
- `pos iMark_anusvara @consonant' <150 0 150 0>`
`iMark_anusvara;`

Positioning

- Entire run visible for Myanmar and USE; implementation dependent for other engines
- Cross-cluster spacing adjustment

Mark width zeroing

- Engines set width of marks to zero
- “Mark” determined by GDEF, not Unicode
- Outlines shifted to left to maintain right margin
- For many marks, that’s what you want
- For some, it isn’t, and you need to compensate using dist feature

OpenType implementations

- Write once, test everywhere
- With complete set of shaping engines
 - DirectWrite, Uniscribe – Windows
 - HarfBuzz – Android, Chrome, Firefox, Java 9
 - CoreText – iOS, macOS

OpenType implementations

- Adobe: incomplete set of shaping engines
 - Brahmic: Devanagari, Bengali, Kannada, Malayalam, Oriya, Tamil, Telugu (v2), Gujarati, Gurmukhi (v1)
- World-Ready Composer – InDesign
- Middle Eastern & South Asian Composer – Photoshop, Illustrator
- Addition of HarfBuzz being explored

OpenType documentation

- <http://www.microsoft.com/en-us/Typography/SpecificationsOverview.aspx>

OpenType tools

- makeotf – compiles feature code into GDEF/GSUB/GPOS tables
- VOLT – visual OpenType layout editor for Windows
- DTL TypeMaster
- Glyphs – complete font development environment

Apple Advanced Typography

Differences to OpenType

- No default behavior – it's all up to the font
- Basic operations: substitution, ligature, insertion, reordering, positioning
- Programmable state machines
- Full access to the complete run; information about line breaks

Where AAT is better

- Support complex script not in Unicode
- Support complex script new in Unicode
- Support scripts too complex for OpenType (Tai Tham)
- Support new characters in existing complex script

Where AAT is better

- Reorder glyphs where OpenType doesn't do it automatically
- Align clusters with margins based on above-or below-base marks
- Cross-cluster substitutions and positioning

Where AAT is no good

- On any non-Apple platform 🙄

Reordering pre-base vowels

- State-action table

	EOT	OOB	B	VPre
StartText	1	1	2	1
SawBase	1	1	2	3

- Action table

	GoTo	MarkFirst?	MarkLast?	Advance?	DoThis
1	StartText	no	no	yes	none
2	SawBase	yes	no	yes	none
3	StartText	no	yes	yes	xD->Dx

Questions?