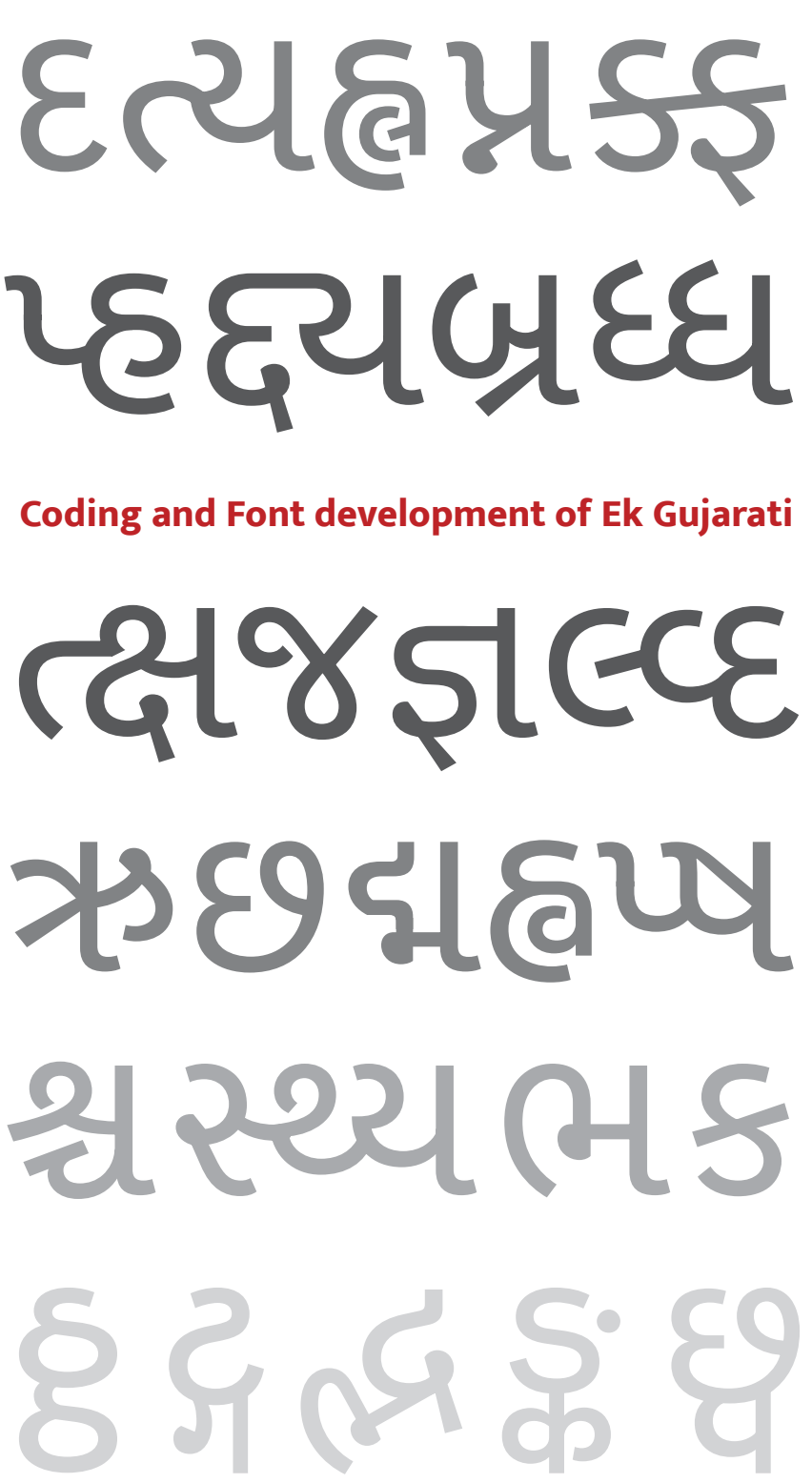


Coding and Font development of Ek Gujarati

Summer Internship Project at Ek Type

Coding and Font development of Ek Gujarati
એક ગુજરાતી નુ કોડીંગ અને ફોન્ટ ડેવલપમેન્ટ

Student Name: Rucha G. Vakhariya
Discipline and Year: Visual Communication 2014-16
Roll Number: 146250009

IDC, IIT Bombay

About the company

Ek Type is a collaborative type design studio based in Mumbai that specialises in developing fonts across all Indian languages, many of which are multi-script.

Their goal is to build one harmonious family across all Indian scripts without letting the visual features of one script dominate over others. This ensures that the fonts can be used successfully for both single and multi-script purposes.

Mentors

Noopur Datye (co-founder Ek Type)

An alumnus of Sir J. J. Institute of Applied Art.
Before joining WhiteCrow, she worked on various design projects at Grandmother India, a design studio based out of Mumbai, India.
Noopur is the designer of Ek Gujarati and was my mentor for this project.

Sarang Kulkarni (co-founder Ek Type)

Also an alumnus of Sir J. J. Institute of Applied art.
In 2005, he founded 'WhiteCrow' – a type foundry and design studio based in Mumbai.
White Crow has steadily found its niche in multi-lingual branding, designing custom typefaces across Indian scripts and calligraphy.

Maithili Shingre (type-designer at Ek Type)

Maithili graduated from Sir J. J. Institute of Applied Art specialising in typography and calligraphy.
She has co-designed 'Modak', a devanagari display typeface under Ek Type and has also worked on 'Ek Mukta'.
She was the one I went to if I had any doubt regarding coding, or when I was stalled by errors while developing the font.

Project outline

Project: Coding and font development
Font: Ek Gujarati
Foundry: Ek Type

The project undertaken was to code and develop Ek Gujarati font to make it compatible with the digital platform.

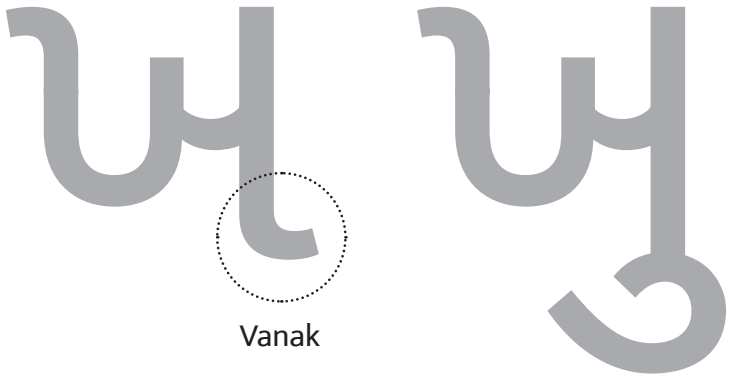
Understanding how coding works in terms of fonts, especially in terms of Indian scripts as they are more complicated than Latin. They have a base glyph and pre-base, post-base, above-base and below-base glyphs that attach to the base glyph.

Getting myself acquainted to the font, font development kit and the different rules used to write the code before developing the font.

Introduction to Gujarati script

Absence of the characteristic horizontal line (shirorekha) running above the letters like Devanagari script.

Distinct feature called ‘Vanak’.



Vanak

It is written from left to right and is not case-sensitive.

This script is mainly used to write Gujarati & Kutchi languages.

Gujarati numeric digits are also different from their devanagari counterparts.

Contemporary Gujarati borrows punctuation from the Latin script such as question mark, exclamation mark, comma, full stop, apostrophe et al.

Introduction to Ek Gujarati

Ek Gujarati is a humanist monolinear typeface.

Part of the Ek multiscript family.

Has 12 swaras and 34 consonants.

The total number of glyphs in Ek Gujarati as of the end of my project is 844.

This includes the swaras, consonants, half forms, conjuncts, akhand conjuncts, rakar forms, matras, ukars, vattu, anuswar, chandrabindu, without vanak forms of all antadanda characters, numbers, punctuation marks and other signs.

Ek Gujarati Regular

અ આ ઇ ઈ ઉ ઊ ઋ એ ઐ ઓ
ઔ અં અઃ

ક ખ ગ ઘ ઙ ચ છ જ ઝ ઞ ટ ઠ ડ
ઢ ણ ત થ દ ધ ન પ ફ બ ભ મ ય
ર લ વ શ ષ સ હ ળ

૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯

ખ ખા ખિ ખી ખુ ખૂ ખે ખૈ ખો ખૌ
ખં ખઃ ખૃ ર્ખ ખૅં ર્ખં

Ek Gujarati Extrabold

અ આ ઇ ઈ ઉ ઊ ઋ એ ઐ ઓ
ઔ અં અઃ

ક ખ ગ ઘ ઙ ચ છ જ ઝ ઞ ટ ઠ ડ
ઢ ણ ત થ દ ધ ન પ ફ બ ભ મ ય
ર લ વ શ ષ સ હ ળ

૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯

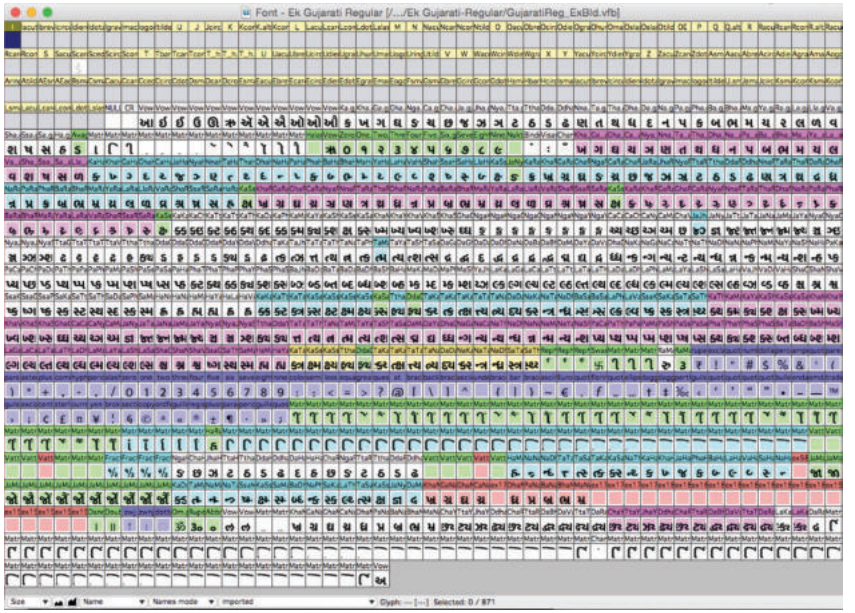
ખ ખા ખિ ખી ખુ ખૂ ખે ખૈ ખો ખૌ
ખં ખઃ ખૃ ર્ખ ખૅં ર્ખં

Editing Glyphs and Coding

Tools used during the project:

FontLab studio 5.1

FontLab studio 5.1 is a fully integrated font editing software to support professional font design requirement.



Adobe Font Development Kit for OpenType (AFDKO)

After installing AFDKO, there are certain commands and rules which are typed out and font development kit automatically generates an opentype font, autohints, checks outlines, checks kernpairs etc.

MakeOTF

MakeOTF builds opentype font.

To use makeOTF, a set of text files are needed, along with true type font files named as font.ttf or font.pfa, arranged in a specific order.

MakeOTF compiles the information in the text files into the font file, and then builds an opentype font.

Three text files needed to make an opentype font:
1. FontMenuNameDB
2. GlyphOrderAndAliasDB
3. features

The names for these text documents is predecided.

FontMenuNameDB

This file contains font menu naming information of a font or a font family. This is how it is written:

```
[PostScriptName]
f= Preferred Family Name
s= Subfamily Name (Style Name)
l= Windows Compatible Menu Name
m=1,Macintosh Compatible Full Name
```

e.g.

```
[AdobeGaramondPro-Semibold]
f=Adobe Garamond Pro
s=Semibold
l=Adobe Garamond Pro Sb
m=1,Adobe Garamond Pro Sb
```

GlyphOrderAndAliasDB (GOADB)

This file contains glyph names, their unicode values and the order of glyphs.

These three columns in the file are are:

1. Glyph name in .vfb file
2. Glyph name to be used to write opentype rules
3. Unicode value specified for the respective glyph like uni0905 is specified for VowelA

It is easier and less confusing when glyph names in .vfb file and glyph names to be used to write rules, are same.

Naming convention

If there are three scripts in a font's glyphset - Devanagari, Bengali, and Gujarati. Then it is better to name the glyph vowel A as VowelA.dv, VowelA.bn, VowelA.gj respectively.

The naming convention that was followed in Ek Gujarati was:

Vovels: VowelA.gj VowelAa.gj VowelI.gj VowelIi.gj and so on

Base glyphs: Ka.gj Kha.gj Ga.gj Gha.gj and so on

Conjuncts: KaKa.gj KhaMa.gj KaTaRa.gj and so on

Without vanak forms: Ka_alt.gj Kha_alt.gj Ga_alt.gj and so on

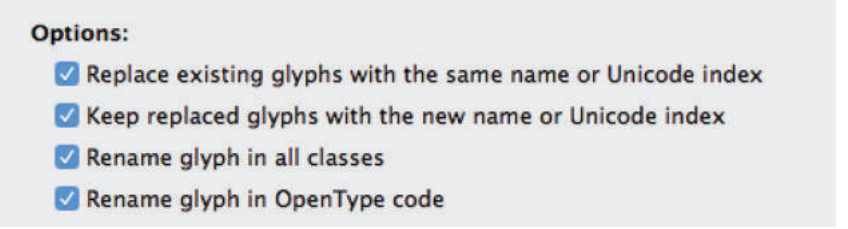
Half forms: KaHalf.gj KhaHalf.gj GaHalf.gj and so on

Matras: MatraA.gj MatraI.gj MatraE.gj and so on

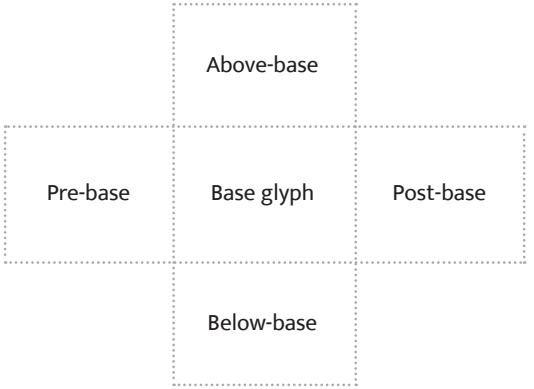
Other signs: Bindu.gj Chandrabindu.gj Halant.gj and so on

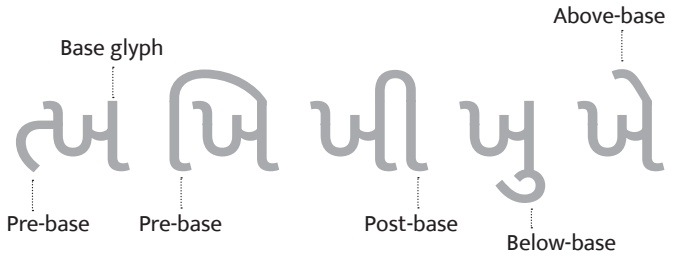
Thus, the GlyphOrderAndAliasDB file looked like this:

```
.notedef  .notedef
space     space
VowelA.gj      VowelA.gj      uni0905
VowelAa.gj     VowelAa.gj      uni0906
<and so on>
```



Features





Some of the features in Latin are: Standard ligatures, Oldstyle/Tabular figures, Superscript-subscript, Small Capitals, Fractions, Stylistic alternates etc.

Indian scripts are complicated than Latin.

There are Pre/Post/Above/Below base glyphs that attach to the base glyph. These can be substituted glyphs or can be positioned glyphs.

The feature file contains:
1. Groups/Classes
2. GSUB (Glyph Substitution)
3. GPOS (Glyph Positioning)
4. GDEF (Glyph Definition Table)

All these text files are compiled in the feature file. The rule is written as:

```
include {Groups};
include {GSUB};
include {GPOS};
include {GDEF};
```

Feature tags

Feature is identified when there is a four letter feature tag.
These feature tags are registered tags.

This is how a feature is written:

```
feature feature_tag {
     # rules or lookups go here
} feature_tag ;
```

Lookups:

Lookups are defined the same way as features.
Lookup names can be decided by us.

An example of a feature and lookup in Ek Gujarati:

```
feature cjct {
lookup pres {
     sub KaHalf.gj Cha.gj by KaCha.gj;
} pres;
} cjct;
```

The order of how features appear in the code is predecided. Opentype features used for Gujarati scripts, are applied in the following order:

Feature tag	Feature function	Layout operation
Localized forms:		
loc	Localization form substitution	GSUB
Basic shaping forms:		
nukt	Nukta form substitution	GSUB
akhn	Akhand ligature substitution	GSUB
rphf	Reph form substitution	GSUB
rkrf	Rakaar form substitution	GSUB
blwf	Below-base form substitution	GSUB
half	Half-form substitution	GSUB
vatu	Vattu variants	GSUB
cjct	Conjunct form substitution	GSUB
Mandatory presentation forms:		
pres	Pre-base substitution	GSUB
abvs	Above-base substitution	GSUB
blws	Below-base substitution	GSUB
psts	Post-base substitution	GSUB
haln	Halant form substitution	GSUB
Positioning features:		
kern	Kerning	GPOS
dist	Distances	GPOS
abvm	Above-base mark positioning	GPOS
blwm	Below-base mark positioning	GPOS

Classes/Groups

When a rule applies to a group of glyphs, a Class is made.

Class name like lookup name is decided by us.

The class name is always preceded by @ sign.

This is how a class is made:

```
@class_name = [ #Glyps go here ];
```

Following is how a class in Ek Gujarati looks:

```
@BelowBaseMarks = [MatraU.gj MatraUu.gj
MatraRu.gj MatraRuu.gj MatraLru.gj
MatraLruu.gj Vattu.gj VattuMatraU.gj
VattuMatraUu.gj VattuMatraRu.gj Halant.gj];
```

Glyph Definition Table (GDEF)

In GDEF, it is defined if a glyph is a simple base glyph, or a mark that will be positioned, or is it a ligature.

Thus, there are three types of glyphs in the GDEF file

1. Simple: a b A B અ આ ક ખ
2. Mark: ઁ ઃ ઼ ઽ
3. Ligature: fi fl (in Latin script)

This is how a GDEF table is written for Ek Gujarati script:

```
@GDEF_simple = [VowelA.gj VowelAa.gj
VowelI.gj VowelIi.gj VowelU.gj VowelUu.gj
VowelRu.gj VowelChadraE.gj and so on];

@GDEF_mark = [Reph.gj RephBindu.gj
RephChandrabindu.gj RaMatraU.gj RaMatraUu.
gj MatraEBindu.gj and so on];

table GDEF {
GlyphClassDef @GDEF_simple,, @GDEF_mark,;
} GDEF;
```

Substitution rules

Substitution rules go in the GSUB file.

Syntax for the substitution is like his:

```
sub target by replacement;
```

e.g. To replace ‘a’ with ‘small caps a’, the rule will be written as

```
sub a by A.sc;
```

One glyph can be substituted by one glyph, many glyphs by one or many by many.

e.g. There are two ways of writing substitution rule for classes:

```
sub [a b c] by [A.sc B.sc C.sc];
```

or

Make classes:

```
@lowercase = [a b c];
@smallcaps = [A.sc B.sc C.sc];
```

and the rule will be written as

```
sub @lowercase by @smallcaps;
```

Contextual substitution

e.g. To replace matraI (Hraswa ikar matra) with a longer matraI when followed by Kha,Sha, the rule will be written like this:

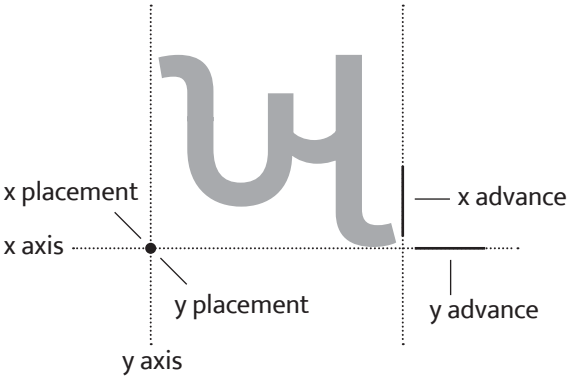
```
@matraI10 = [Ka.gj Sha.gj];
sub MatraI’ @matraI10 by MatraI10.gj;
```

Without contextual substitution

With contextual substitution

Positioning rules

Position rules go in the GPOS file.
To position a mark or simple glyph, the coordinate system is used.



The values are written like this

`<xPlacement yPlacement xAdvance yAdvance>`
e.g. `<10 20 0 0>`

This is called a ‘value record’

Modification of the origin (placement) point of the glyph shifts that glyph. Modification of the advance shifts the next glyph.

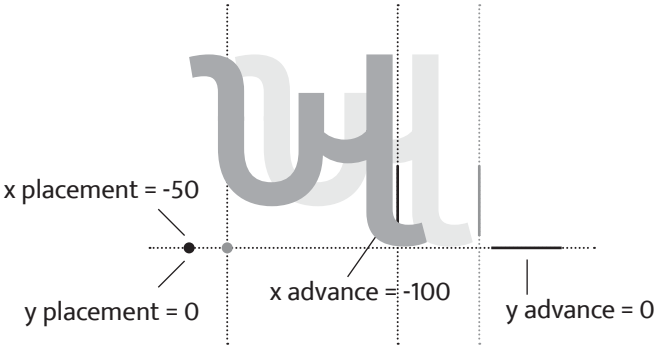
Single positioning

A single positioning rule is written as:

```
pos <glyph/glyphclass> <value record>;
```

e.g. To reduce left and right bearings of a glyph each by 50 units, the rule will be:

```
pos Kha <-50 0 -100 0>;
```



To shift a glyph by 100 units

```
pos Kha <0 100 0 -100>;
```

Kerning/ Pair positioning

FontLab generates these codes for us. We just have to copy them and paste them in the GPOS file.

These rules are put under the ‘dist’ feature instead of the ‘kern’ feature as the ‘dist’ feature does not rely on the application to enable kerning.

Metrics and Kerning classes

The distances between the extreme ends of each side of the glyph including the allocated space are the ‘sidebearings’.

Certain glyphs like ક ફ can share the same bearings, so they will fall under the same metrics class.

Conjuncts that end with છ can fall under the same metrics class.

Metrics classes help make life easier, as one doesnt have to individually change every glyphs bearing. Changing bearing of one glyph of a group changes the bearing of all the others in the group automatically.

Kerning classes work in a similar way. For eg: Glyphs with ગ અ ઝ on the left will fall under the same kerning class. This will help when the kerning of one of these glyphs is changed, it will automatically apply to others.

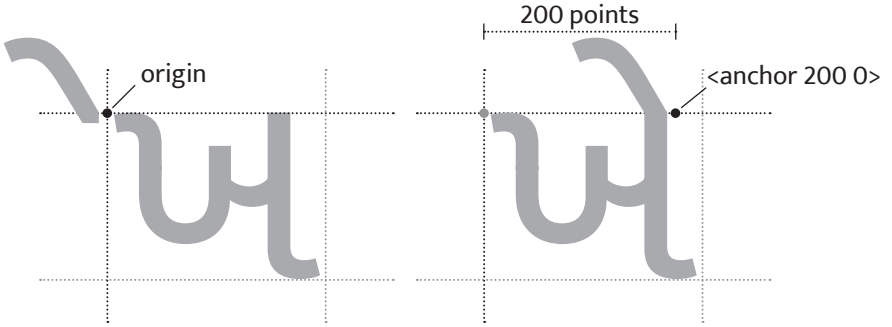
Mark to Base attachment positioning

This type of positioning rule is used to position mark glyphs to simple base glyphs.

e.g. If MatraE is to be positioned above Kha then rule will be written as:

```
@abovebase = [MatraE.gj MatraAi.gj];
@Kha = [Kha.gj];

pos base @Kha <anchor 200 0> mark @
abovebase;
```



Multiple mark glyph classes can also be positioned to one base glyph as follows:

e.g. If there are two mark classes:

1. `@abovebase_1 = [MatraE.gj MatraAi.gj] ;`
2. `@abovebase_2 = [Bindu.gj Chandrabindu.gj];`

and we want all of them to position with 'Kha'. The rule will be written like this:

```
pos base Kha.gj
     <anchor 200 0> mark @abovebase_1
     <anchor 210 0> mark @abovebase_2;
```

Language system

For features to work, the script and language in which those features are going to work need to be specified.

This is how it is written for Gujarati:

```
languagesystem DFLT dflt;
languagesystem latn dflt;
languagesystem gujr dflt;
languagesystem gujr GUJ ;
languagesystem gjr2 dflt;
languagesystem gjr2 GUJ ;
```

This syntax are present before any feature definitions. Once this is written all these scripts and languages get applied to all features and there rules.

Using makeOTF to generate opentype font

All files go in a folder in a specific order, which is called a font directory. This is what a font directory contains:

1. Generated ttf named font.ttf or font.pfa
2. Features
3. GlyphOrderAndAliasDB
4. FontMenuNameDB
5. Groups
6. GSUB
7: GPOS
8. GDEF
9. .vfb file

To generate font:

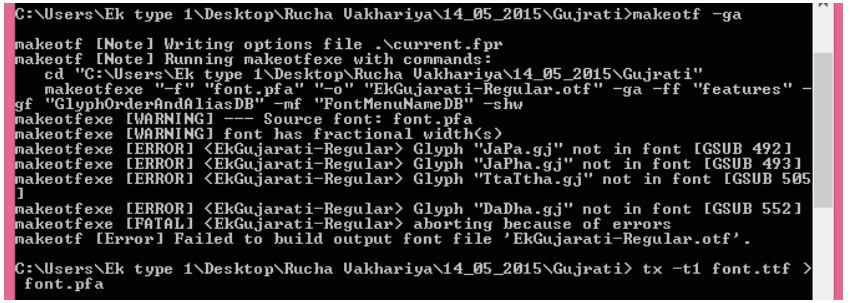
On a windows system open the command prompt window (right click and choose Run as Administrator)
and type the following:

```
cd<space>path to the font directory<enter>
tx -t1 font.ttf > font.pfa<enter>
makeotf -ga<enter>
```

If there are no errors the font will successfully get generated. The command prompt window should give a message like this:

MakeOTF errors

Most of the times these errors are self explanatory. They also specify where exactly is the problem occuring.



Some things I learned to avoid errors:

1. Different rules should not go under same lookup.

2. All lookups should be assigned to a feature.

3. The order within a feature should be such that the longest substitution must go first. For eg: four letter conjunct substitution rule will be written before three letter conjuncts followed by two lettered conjuncts.

4. When substituting classes/groups, the glyph order and number of glyphs should be the same in both the target and replacement classes.

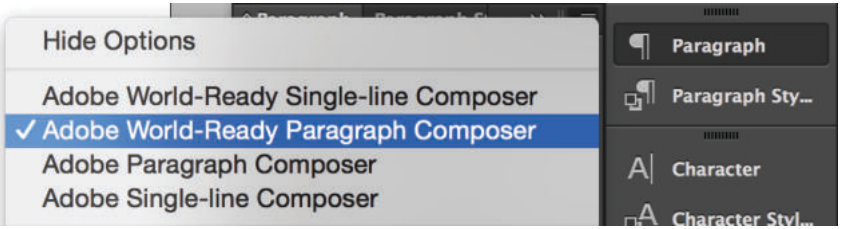
5. Missing glyphs. This error appears when there is a spelling mistake in the glyph name in one of the files or it is missing from either GDEF or GlyphAliasAndDB file.

6. Small mistakes like missing { } [ ] ; etc can also cause errors.

Font testing

The software I used for testing was Adobe Indesign CC.

Indesign CC provides the option of World-ready composer, which makes it compatible for using Indic scripts. Once this is enabled, all rules work fine.



Thank you

ધન્યવાદ