ucharclasses

Mike “Pomax” Kamermans

September 25, 2012

Contents

1 Introduction

Sometimes you don't want to have to bother with font switching just because you're using languages that are distinct enough to use different Unicode blocks, but aren't covered by the polyglossia package. Where normal word processing packages such as MS, Star- or OpenOffice pretty much handle this for you, LaTeX (because it needs you to tell it what to do) has no default behaviour for this, and so we arrive at a need for a package that does this for us. You already discovered that regular LaTeX has no understanding of Unicode (in fact, it has no understanding of 8-bit characters at all, it likes them in seven bits instead), and ended up going for Xe(La)TeX as your TeX compiler of choice, which means you now have two excellent resources available: fontspec, and ucharclasses.

The first of these lets you pick fonts based on what your system calls them, without needing to rewrite them as MetaFont files. This is convenient. This is good. The second lets you define what should happen when we change from a character in one Unicode block to a character in another. This is also convenient, and paired with fontspec it offers automatic fontswitching in the same way that normal Office applications take care of this for you. With one big difference: you stay in control. In an Office application, if at some point you need the switch rule to use a completely different rule, that's just too bad for you. In Xe(La)TeX, you stay on top of things and still get to say exactly what happens, and when.

For instance, this document has no explicit font codes in the text itself. Instead, there are a few Unicode block transition rules defined, which all say "when entering block ..., use fontspec to change the font to ...". As such, typesetting the following list in the appropriate fonts just works:

- English: This is an English phrase (using Palatino Linotype)
- Japanese: 日本語が分かりますか (using Ume Mincho)
- Thai: คุณพูดภาษาอังกฤษได้ไหม (using IrisUPC)
- Sinhala: කරුණාකරල ඒක නැවත කියන්න පුළුවන්ද (using Iskoola Pota)
- Malayalam: നിങ്ങളുടെ പരേന്താണ്? (using Arial Unicode MS)
- and even domino tiles, 🁢 🁣 🁤 🁥 🁦 🁧, and mahjong tiles: 🀅 (using FreeFont)

However, be aware that this only "just works" for Unicode blocks. If you are working with typographically overlapping languages, such as combining English and Vietnamese in one document, things get a lot more complex if you want one font for English and another for Vietnamese. Both of these languagese use Latin blocks, so it is inherently impossible to tell which language is intended based on which Unicode block a character in a word belongs to.

As an example, this document uses one rule for applying a font for general CJK, and an override with a different font for all Japanese-specific CJK characters. This causes a problem for Chinese, because both Japanese and Chinese mostly use characters from the "CJK Unified Ideographs" block, but most Japanese fonts contain fewer characters than are necessary to typeset Chinese:

- Chinese, using the Japanese CJK font, which may have gaps: 我的母. 是. (uses Ume Mincho, which does not contain the three Chinese-specific characters used in that phrase)

We can get around this by explicitly setting the font to one that supports Chinese, turning off the switching rules for the stretch of Chinese text, using {\uccoff + a fontspec rule + the text we wanted to typeset + \uccon}. This gives us: 我的母语是汉语 (This now explicitly uses Han Nom A).

2 Use

In order to get this all to work, the only thing that had to be incidated was a set of transition rules in the preamble:

```
\usepackage[CJK, Latin, Thai, Sinhala, Malayalam,
            DominoTiles, MahjongTiles]{ucharclasses}
\usepackage{fontspec}
\usepackage{bidi}

\setDefaultTransitions{\fontspec{Code2000}}{}
\setTransitionsForLatin{\fontspec{Palatino Linotype}}{}
\setTransitionsForCJK{\fontspec{HAN NOM A}{}
\setTransitionsForJapanese{\fontspec{Ume Mincho}}{}

\setTransitionTo{Thai}{\fontspec{IrisUPC}}
\setTransitionTo{Sinhala}{\fontspec{Iskoola Pota}}
\setTransitionTo{Malayalam}{\fontspec{Arial Unicode MS}}
\setTransitionTo{DominoTiles}{\fontspec{FreeSerif}}
\setTransitionTo{MahjongTiles}{\fontspec{FreeSerif}}
```

By default, ucharclasses is agnostic with regard to what you want inserted at the start or end of Unicode blocks, so while using this package for font switching is the most obvious application, you could also use it for far more creative purposes.

2.1 Overriding ucharclass transitions

If you need to "override" ucharclass transition rules (for instance, you want a custom font for a bit of cross-Unicode-block text), you will want to temporarily disable and reenabled XeTeX's interchartoks state. You can do this in three ways:

1. call [\XeTeXinterchartokstate = 0] before, and [\XeTeXinterchartokstate = 1] after you're done,
2. call the macros \disableTransitionRules before, and \enableTransitionRules after you're done, or

3. call \uccoff before, and \uccon after you're done.

This last option is mainly there because it's nice and short, and is more convenient in a scoped environment {\uccoff such as this\uccon} where you only want to override the transition behaviour within a paragraph. If you need it disabled for a few blocks of text instead, the full name commands are probably a better choice, because it makes your .tex more readable. As the base XeTeX command uses the unLaTeXy "... = ..." construction, it's best to avoid it outside of the preamble (and when using ucharclasses, should not be in the preamble at all).

3 Problems with RTL languages

The overlapping block problem is especially notable when using RTL/LTR rules for languages such as Arabic or Hebrew. While you would want to be able to specify something along the lines of:

\setTransitionsForArabics{\fontspec{Tahoma}\setRTL}{\setLTR}

this will not work, because Arabic (and Hebrew, and other RTL languages) has things like spaces in it, and so rather than ending with a full sentence that starts with \setRTL, then the Arabic text, and then finally \setLTR, every word in the Arabic sentence will be wrapped by \setRTL and \setLTR, effectively getting the typesetting all wrong, because going from Arabic to a space character "leaves" the Arabic block, so the transition rule for leaving the Arabic block is applied.

If you need script support, rather than Unicode blocks, you may want to have a look at the polyglossia package instead. You can try to combine the two packages by relying on \uccoff and textbackslash uccon to turn off Unicode block transitions inside regions of text, but this may not always work, or may have interesting interaction side-effects.

4 Commands

4.1 \setTransitionTo[2]

This command has two arguments:

1. The name of the Unicode class to which the transition should apply (see 'Unicode blocks' list)
2. The code you want used when entering this Unicode block

4.2 \setTransitionFrom[2]

This command has two arguments:

1. The name of the Unicode class to which the transition should apply (see 'Unicode blocks' list)
2. The code you want used when exiting this Unicode block

4.3 \setTransitions[3]

This command has three arguments:

1. The name of the Unicode class to which the transition should apply (see 'Unicode blocks' list)
2. The code you want used when entering this Unicode block
3. The code you want used when exiting this Unicode block

4.4 \setTransitionsForXXXX[2]

There are a number of these commands, pertaining to particular "informal groups": collections of Unicode blocks which can be considered part of a single meta-block. Available informal groups (the names of which replace the XXXX in the section-stated command) are:

- Arabics
- Chinese
- CJK
- Cyrillics
- Diacritics
- Greek
- Korean
- Japanese
- Latin
- Mathematics
- Phonetics
- Punctuation
- Symbols
- Yi

Furthermore, these commands have two arguments:

1. The code you want used when entering blocks from the command's informal group
2. The code you want used when exiting blocks from the command's informal group

4.5 \setDefaultTransitions[2]

This is a blanket command that lets you set up the same to and from transition rules for all blocks in one go. It has (fairly obviously) two arguments:

1. The code you want used when entering any Unicode block
2. The code you want used when exiting any Unicode block

5 Code

The code relies on running through individual definition blcosk for each Unicode blocks, conditioned to whether ucharclasses is loaded with package options or not:

```
...
\ifboolexpr{
 togl {@loadAll} or togl {@loadBasicLatin}
}{\@defineUnicodeClass{\BasicLatinClass}{32}{127}}
...
```

The classes are automatically numbered by using the \newXeTeXintercharclass command, and every time a new class is defined, the class counter goes up. After all desired classes have been defined, the code iterates over the class numbers from lower bound to upper bound.

The block loading code is defined as follows:

```
\newcounter{glyphcounter}
\newcommand{\@defineUnicodeClass}[3]{
    \newXeTeXintercharclass#1
    \forloop{glyphcounter}{#2}{\value{glyphcounter}<#3}{
        \XeTeXcharclass\value{glyphcounter}=#1}
    \XeTeXcharclass#3=#1}

\newXeTeXintercharclass\@classstart
...
\ifboolexpr{
 togl {@loadAll} or togl {@loadBasicLatin}
}{\@defineUnicodeClass{\BasicLatinClass}{32}{127}}
...
\newXeTeXintercharclass\@classend
```

And the transition commands are defined as follows:

```
\newcommand{\setTransitionsFor}[3]{
  \forloop{iclass}{he\@classstart}{\value{iclass} < \@nameuse{#1Class}}{
     \@transition{he\value{iclass}}{\@nameuse{#1Class}}{{#2}}
  \@transition{\@nameuse{#1Class}}{he\value{iclass}}{{#3}}}
\addtocounter{iclass}{2}
\forloop{iclass}{\value{iclass}}{\value{iclass} < he\@classend}{
  \@transition{he\value{iclass}}{\@nameuse{#1Class}}{{#2}}
  \@transition{\@nameuse{#1Class}}{he\value{iclass}}{{#3}}}
% and a binding for the transitions to and from boundary characters
\@transition{255}{\@nameuse{#1Class}}{{#2}}
\@transition{\@nameuse{#1Class}}{255}{{#3}}}

\newcommand{\setTransitionTo}[2] {
\forloop{iclass}{\the\@classstart}{\value{iclass} < \@nameuse{#1Class}}{
  \@transition{\the\value{iclass}}{\@nameuse{#1Class}}{{#2}}}
\addtocounter{iclass}{2}
\forloop{iclass}{\value{iclass}}{\value{iclass} < \the\@classend}{
  \@transition{\the\value{iclass}}{\@nameuse{#1Class}}{{#2}}}
```

```
  % and a binding for the transition from boundary characters
  \@transition{255}{\@nameuse{#1Class}}{{#2}}}

 \newcommand{\setTransitionFrom}[2]{
  \forloop{iclass}{\the\@classstart}{\value{iclass} < \@nameuse{#1Class}}{
    \@transition{\@nameuse{#1Class}}{\the\value{iclass}}{{#2}}}
  \addtocounter{iclass}{2}
  \forloop{iclass}{\value{iclass}}{\value{iclass} < \the\@classend}{
    \@transition{\@nameuse{#1Class}}{\the\value{iclass}}{{#2}}}
  % and a binding for the transition to boundary characters
  \@transition{\@nameuse{#1Class}}{255}{{#2}}}
```

The broad level \setTransitionsFor(InformalGroupName)[2] commands are essentially wrapper commands, calling \setTransitionsFor for each blocks that is in the informal group. For Arabic, for instance, the code is:

```
\newcommand{\setTransitionsForArabic}[2]{
  \setTransitionsFor{Arabic}{#1}{#2}
  \setTransitionsFor{ArabicPresentationFormsA}{#1}{#2}
  \setTransitionsFor{ArabicPresentationFormsB}{#1}{#2}
  \setTransitionsFor{ArabicSupplement}{#1}{#2}
}
```

6 Package options and Unicode blocks

The following Unicode blocks are available for use in transition rules, as well as for use as package options when you want ucharclasses to only load those classes that you know are used in your document:

- AegeanNumbers
- AlphabeticPresentationForms
- AncientGreekMusicalNotation
- AncientGreekNumbers
- AncientSymbols
- Arabic
- ArabicPresentationFormsA
- ArabicPresentationFormsB
- ArabicSupplement
- Armenian
- Arrows
- Balinese
- BasicLatin
- Bengali
- BlockElements
- Bopomofo
- BopomofoExtended
- BoxDrawing
- BraillePatterns
- Buginese
- Buhid
- ByzantineMusicalSymbols
- Carian
- Cham
- Cherokee
- CJKCompatibility
- CJKCompatibilityForms
- CJKCompatibilityIdeographs
- CJKCompatibilityIdeographsSupplement
- CJKRadicalsSupplement
- CJKStrokes
- CJKSymbolsandPunctuation

- CJKUnifiedIdeographs
- CJKUnifiedIdeographsExtensionA
- CJKUnifiedIdeographsExtensionB
- CombiningDiacriticalMarks
- CombiningDiacriticalMarksforSymbols
- CombiningDiacriticalMarksSupplement
- CombiningHalfMarks
- ControlPictures
- Coptic
- CountingRodNumerals
- Cuneiform
- CuneiformNumbersandPunctuation
- CurrencySymbols
- CypriotSyllabary
- Cyrillic
- CyrillicExtendedA
- CyrillicExtendedB
- CyrillicSupplement
- Deseret
- Devanagari
- Dingbats
- DominoTiles
- EnclosedAlphanumerics
- EnclosedCJKLettersandMonths
- Ethiopic
- EthiopicExtended
- EthiopicSupplement
- GeneralPunctuation
- GeometricShapes
- Georgian
- GeorgianSupplement
- Glagolitic
- Gothic
- GreekandCoptic
- GreekExtended
- Gujarati
- Gurmukhi
- HalfwidthandFullwidthForms
- HangulCompatibilityJamo
- HangulJamo
- HangulSyllables
- Hanunoo
- Hebrew
- Hiragana
- IdeographicDescriptionCharacters
- IPAExtensions
- Kanbun
- KangxiRadicals
- Kannada
- Katakana
- KatakanaPhoneticExtensions
- KayahLi
- Kharoshthi
- Khmer
- KhmerSymbols
- Lao
- LatinExtendedAdditional
- LatinExtendedA
- LatinExtendedB
- LatinExtendedC
- LatinExtendedD
- LatinSupplement
- Lepcha
- LetterlikeSymbols
- Limbu
- LinearBIdeograms
- LinearBSyllabary
- Lycian
- Lydian
- MahjongTiles
- Malayalam
- MathematicalAlphanumericSymbols
- MathematicalOperators
- MiscellaneousMathematicalSymbolsA
- MiscellaneousMathematicalSymbolsB
- MiscellaneousSymbols
- MiscellaneousSymbolsandArrows
- MiscellaneousTechnical
- ModifierToneLetters
- Mongolian
- MusicalSymbols
- Myanmar
- NewTaiLue
- NKo
- NumberForms
- Ogham
- OldChiki
- OldItalic
- OldPersian
- OpticalCharacterRecognition
- Oriya
- Osmanya

- PhagsPa
- PhaistosDisc
- Phoenician
- PhoneticExtensions
- PhoneticExtensionsSupplement
- PrivateUseArea
- Rejang
- Runic
- Saurashtra
- Shavian
- Sinhala
- SmallFormVariants
- SpacingModifierLetters
- Specials
- SuperscriptsandSubscripts
- SupplementalArrowsA
- SupplementalArrowsB
- SupplementalMathematicalOperators
- SupplementalPunctuation
- SupplementaryPrivateUseAreaA
- SupplementaryPrivateUseAreaB
- SylotiNagri
- Syriac
- Tagalog
- Tagbanwa
- Tags
- TaiLe
- TaiXuanJingSymbols
- Tamil
- Telugu
- Thaana
- Thai
- Tibetan
- Tifinagh
- Ugaritic
- UnifiedCanadianAboriginalSyllabics
- Vai
- VariationSelectors
- VariationSelectorsSupplement
- VerticalForms
- YiRadicals
- YiSyllables
- YijingHexagramSymbols

In addition, the informal blocks for use as package option are:

- Arabics
- Chinese
- CJK
- Cyrillics
- Diacritics
- Greek
- Korean
- Japanese
- Latin
- Mathematics
- Phonetics
- Punctuation
- Symbols
- Yi