luaTEX & Unicode

GUT 2007

Fabrice Popineau

avec beaucoup d'aide de

Hans Hagen

un peu d'histoire

nécessité de rendre TEX plus programmable qu'il ne l'est

Perl, Ruby, Python ... et maintenant
Lua (inspiré par l'expérience de SciTE)

défié par Hans Hagen, Hartmut Henkel a intégré Lua dans TEX

et Taco Hoekwater a rejoint le projet de réaliser luaTEX

à la même période, le projet Oriental TEX (de Idris Samawi Hamid) avait besoin d'un remplaçant pour Aleph

lorsque le financement du développement devint possible, le projet a vraiment démarré

Taco s'occupe du code interne, Hans du code Lua, le développement est très rapide

quelques uns des objectifs

les données internes passeront sur 21--32 bits

UTF-8 en entrée et en interne

nécessité d'un support flexible pour les polices OpenType

intégrer certaines capacités d'Aleph (mise en page bidirectionnelle à la place de celle de e-TEX)

accéder à la plupart des structures de données internes et des procédures de TEX au travers de callbacks

nouvelles extensions de TEX (maths par exemple)

intégration de MetaPost dans TEX

comment le projet s'articule

pdfTeX : 1++ sera figé (seulement correction de bugs)

luaTeX : est de facto pdfTeX 2++

TeX-Gyre : ce projet fournira les polices

mplib : MetaPost transformé en bibliothèque

MKI, MKII, MkIV : la prochaine génération de ConTeXt (l'environnement de test)

à quoi sert Lua ?

l'utilisateur peut appeler directement Lua

il y a un programme binaire indépendant (texlua)

les développeurs peuvent étendre ou
remplacer des fonctionnalités existantes

les nouvelles fonctionnalités sont écrites exclusivement en Lua

sauf au bas niveau où le code est écrit en
C (par exemple le chargement des polices)

ou en est luaTeX

disponible :

la commande \directlua

la gestion des catcodes

la possibilité de redéfinir les I/O de fichier

des procédures de chargement de tfm et de polices OpenType

des points d'entrée dans le tokenizer

accès à des gestionnaires de noeuds et aux attributs des noeuds

de nouvelles fonctionnalités (presque) tous les jours...

mais :

pas encore de recette toute faite !

plans pour la suite

les procédures de construction de paragraphe et de page seront ouvertes

le support des maths sera également ouvert et étendu

la gestion des erreurs sera paramétrable

le code existant sera nettoyé (éventuellement remplacé)

en dernier, les composants du noyau de TEX kernel seront réécrits en C

au final, le code Lua connectera tous les composants de TEX

LuaTeX ne comprend que le codage UTF en entrée

LuaTeX fournit des mécanismes pour que les utilisateurs puissent écrire leur propres routines de conversions vers UTF

LuaTeX assume de l'UTF-8 en entrée, mais doit être capable de traiter de l'UTF-16 (et UTF-32 ?)

notion de combinaison de caractères : un `a` suivi de `'` devient `à`.

modification de LUA pour disposer de fonctions d'itération rapide sur une chaîne (version UTF-8 multibyte, ou version UTF-16)

support de UTF-16 : complexité supplémentaire due à l'ordre des octets. Difficulté de repérer les fins de lignes (combinaison des caractères 10 et 13).

passage à UTF

autres codages d'entrée : il est assez simple d'itérer sur une chaîne d'octets en entrée et de remplacer les valeurs trouvées par leur contrepartie en UTF. Il suffit de maintenir des tables de conversion.

deux endroits pour introduire des convertisseurs :

1. lecture d'une ligne dans un fichier
2. le callback `process_input_buffer` appelé chaque fois que TEX a besoin d'une nouvelle ligne.

on peut surcharger les fonctions d'ouverture et de lecture pour un fichier, donc accrocher le mécanisme de combinaison à cet endroit. Ce mécanisme est ignoré pour UTF-16

passage à UTF

Quand on a un fichier en UTF ouvert,
on lit ligne par ligne dans ce fichier en
combinant les caractères lorsque c'est autorisé

Quand LuaTeX ouvre un fichier, on regarde à l'aide des premiers octets si c'est de l'UTF-16, et l'ordre des octets. Si c'est de l'UTF-16, le fichier est lu en mémoire, converti en UTF-8, les lignes sont repérées et on fournit un lecteur qui va retourner une ligne après l'autre en UTF-8

Quand on doit recoder l'entrée (utilisation d'un régime d'entrée en ConTeXt), le lecteur normal est utilisé, l'entrée est recodée au vol sans combinaison de caractères.

Utilisation d'une astuce pour écrire en 8 bits purs dans le DVI ou le PDF reposant sur un vecteur de valeur en haut de la table allouée à Unicode(256 valeurs à partir de `0x110000`). Le décalage est soustrait lors de l'écriture effective dans le fichier.

traitement UTF en résumé

OpenType, Latin Modern et TEX Gyre ... mais
quand on a beaucoup de polices Type I ?

Première étape : réécrire des procédures
de lecture pour les fichiers vf et tfm

Fusion de la lecture des tfm et des ofm : le mécanisme
de gestion des polices de Aleph a déjà été importé

Les fichiers tfm sont lus sous forme
de tables accessibles depuis Lua

Lecture des afm : en principe, on peut se passer
des tfm si on a les afm et un vecteur d'encodage

En fait on pourra aussi se passer des fichiers map :-)

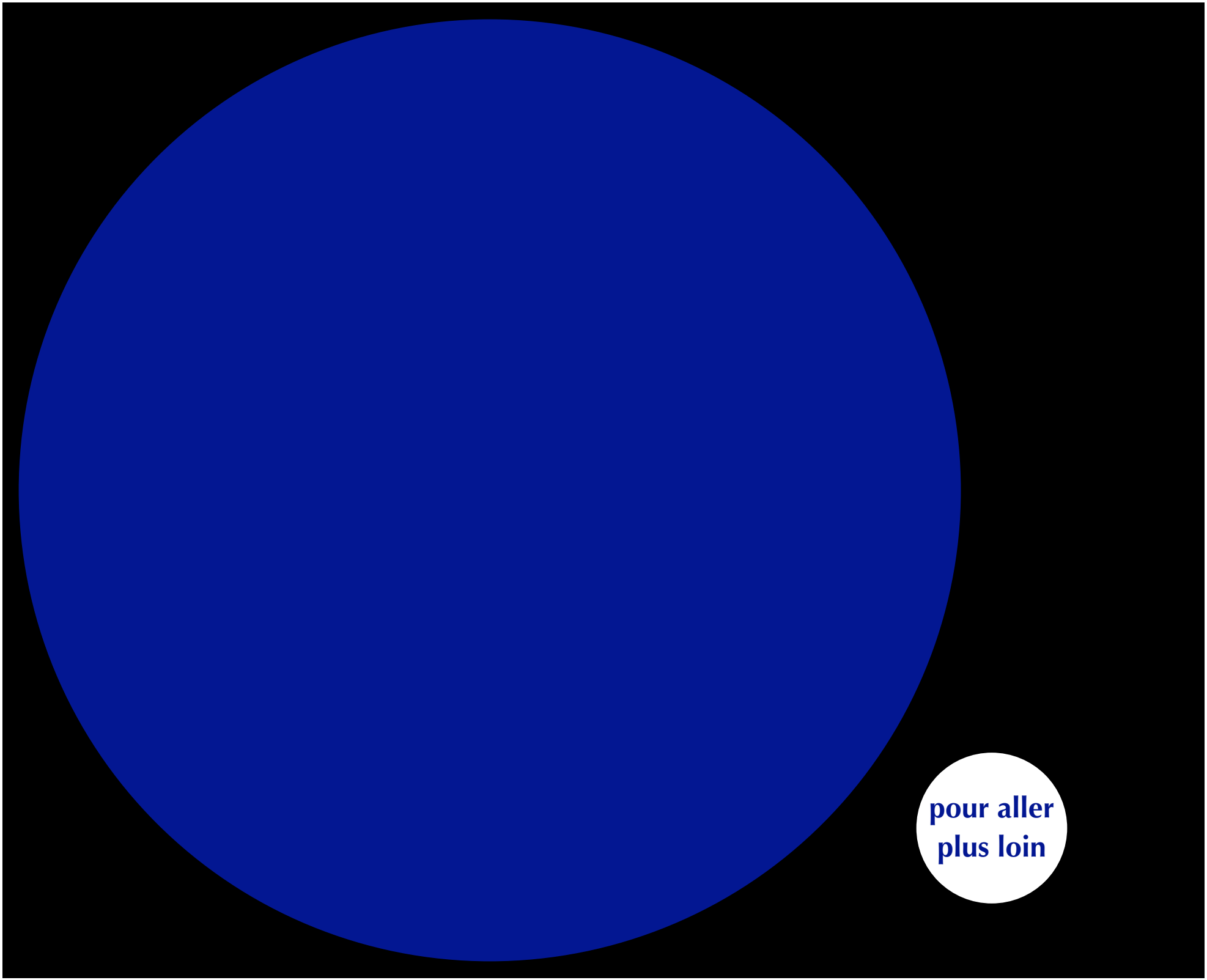
nouvelle vision des polices

OpenType et TTF OpenType et les « features »

Possibilité de fabriquer des polices virtuelles « à la volée »

Codage interne UTF-8 : on peut se passer des encodages, l'association se fait en unicode

nouvelle vision des polices (2)


pour aller
plus loin

beaucoup de variantes de tirets !

menus déroulants pour des symboles :
le α grec n'est pas le α mathématique !

il y a une astuce dans LuaTeX pour obtenir du 8bit dans la sortie PDF (utilisation d'un vecteur privé en Unicode)

il y a des éditeurs de texte qui ne savent pas gérer les ligatures correctement (f+i = fi devient un seul caractère)

... c'est pourquoi il est bien utile d'avoir un outil programmable comme LuaTeXpour se débrouiller des situations les plus difficiles !

caveats