



Rendu des polices et performances


Le 15 septembre 2008 à 03:29.

En essayant de comprendre les problèmes de performances que je peux rencontrer avec l'ami FireFox, j'ai eu une petite surprise en explorant la voie "GTK". En utilisant l'outil gtkperf, je me suis ainsi rendu compte que sur les 8 secondes du test, 2.24 étaient passées sur la partie "Text".


Brève histoire de police

Il y a près de 30 ans, à l'époque du roi [CGA](#)^[1] que beaucoup ne doivent plus connaître, les polices étaient de la compétence du mode texte. Les caractères alors étaient consignés dans d'exiguës petite matrices de 8x8 pixels sur 25 lignes de 80 colonnes.

6 ans plus tard, avec la norme VGA, nous avons toujours un mode texte, mais avec des caractères plus fins dans des matrices de 9x16 pixels. On pouvait même "uploader" de nouvelles polices dans le contrôleur hardware, bonheur 😊

 Puis, fin des années 80, vint une guerre entre Adobe et Apple, chacun avec son format de police pour les imprimantes [postscript](#)

. Pour Adobe nous avons [Type 1](#)^[2], et pour Apple, le [TrueType](#)^[3]. Là, plus rien à voir avec nos polices Bitmap, les deux systèmes travaillaient sur des polices vectorielles, indépendante du support (papier ou écran), lissées (anti-aliasing) et utilisant des techniques d'optimisation des contours (font hinting) permettant, en fonction du support, de déterminer quels points doivent ou pas être dessinés. Ce dernier aspect est fondamental car c'est lui qui rend une fonte lisible dans de petites résolutions. Apple pour même jusqu'à incorporer dans le système TrueType une petite machine virtuelle dédiées à cet usage. Les concepteurs des polices pouvaient ainsi ajouter de petits programmes embarqués dans le fichier pour optimiser de manière dynamique le rendu. Rapidement ces technologies ne se limitent plus à l'impression et chacun propose son propre moteur d'affichage, TrueType System pour Apple et ATM pour Adobe.

 Microsoft, dont on avait pas encore entendu parler, va recevoir une licence de TrueType de la part d'Apple. Officiellement en échange d'une licence sur [TrueImage](#)^[4], officieusement pour dégommer Adobe bien trop coriace dans le mode de l'édition. Du coup, c'est donc en 91, que Microsoft ajoute TrueType pour la première fois, à Windows 3.1. Plus tard la technologie est améliorée pour obtenir ClearType, une implémentation d'un système déjà connu depuis l'Apple II, l'utilisation des sous-pixels (SubPixel Rendering). L'idée de base de cette technologie est d'améliorer encore la qualité de rendu des polices à l'affichage en utilisant le fait que chaque pixel d'un écran est en réalité composé de 3 sous-pixels de 3 couleurs différentes (rouge, vert, bleu). Sachant cela, il devient possible en monochrome, de générer des polices trois fois plus fines avec comme effet indésirable, les petits artefacts colorés que nous connaissons tous.

Enfin, quelques années plus tard arrive [FreeType](#)^[5], le projet d'un moteur libre de rendu des fontes TrueType et Type 1. Tellement compatible qu'il n'en était pas libre, car il implémentait le fameux moteur d'optimisation d'Apple [sans en avoir la licence](#)^[6]... Les développeurs ont vite corrigé le tir en inventant purement et simplement [leur propre technique d'optimisation des contours](#)^[7], l'Auto-Hinter, permettant de faire le même travail, sans l'utilisation du vilain langage. Et le résultat, ben vous l'avez sûrement sous les yeux 😊

Performances

Ce que nous permet de comprendre cette petite rétrospective est que le rendu des polices est un travail mathématiquement sérieux. Entre la conversion des formats vectoriels, l'optimisation des contours et l'adoucissement des courbes, la version numérique de la machine de [Gutenberg](#)^[8] n'est pas une mince affaire. Cela explique déjà pourquoi sur un teste de 8.82 secondes, 2.31, soit **plus d'un quart du temps** est pris par le test du texte.

Maintenant voyons ce que cela donne en faisant varier les options (sous Gnome, panneau Système/Préférences/Apparence/Polices/Détails) :

Lissage/Optimisation	Aucun	Léger	Moyen	Total
Aucun	0.44	0.44	0.44	0.44
Classique	0.48	0.46	0.50	0.49
Sous-Pixels	2.40	2.42	2.20	2.24

Le premier constat est que le moteur d'optimisation n'induit aucun résultat notable. Plus étonnant, que la fonte soit adoucie ou pas ne change rien non plus. A bien y réfléchir ce n'est pas si étonnant que cela car FreeType (comme ses collègues) ne dessine jamais les glyphes directement à l'écran. Il les converti d'abord en bitmap monochrome ou niveaux de gris selon que le lissage est activé ou pas, et stocke cela dans un cache.

En revanche, l'activation des sous-pixels induit un résultat 5.5 fois plus lents qu'un adoucissement classique. Il est là notre "problème". Une explication possible est déjà que les bitmaps générés sont logiquement beaucoup plus volumineux que leur version par biveaux de gris. En effet, si l'on imagine qu'un caractère occupe une matrice de 14x14=196 pixels, en mode

sous-pixels, le même caractère s'étendra sur $(3*14)*(3*14)$ pixels, soit un bitmap 9 fois plus volumineux.

Conclusion

Cette "découverte" a bien arrangé les choses concernant les performances tristes performances de FireFox lors du scroll de certaines pages. En tout cas, il est certain que le gain de vitesse est sensible pour une différence de rendu que je trouve personnellement assez discutable.

*<http://artisan.karma-lab.net/node/1635>
(C) artisan numerique - CC BY-SA*

Liens:

[1] <http://fr.wikipedia.org/wiki/CGA>

[2] <http://fr.wikipedia.org/wiki/postscript>

[3] <http://fr.wikipedia.org/wiki/TrueType>

[4] <http://fr.wikipedia.org/wiki/TrueImage>

[5] <http://www.freetype.org/index2.html>

[6] <http://www.freetype.org/patents.html>

[7] <http://freetype.sourceforge.net/autohinting/index.html>

[8] <http://fr.wikipedia.org/wiki/Gutenberg>