État de HTML5 <canvas> dans QtWebKit

Qt by Nokia

Par Andreas Kling - traducteur : Jonathan Courtois - Qt Labs

Date de publication : 8 septembre 2010

Dernière mise à jour : 31 octobre 2010

L'HTML5 introduit un nouvel élément de bloc nommé <canvas> pour vos pages web. Cet article présente rapidement ses conformités et performances ainsi que de nombreuses démos. Il est important de noter que cet article concerne la branche trunk actuelle de QtWebKit, c'est pourquoi certaines parties ne sont pas pertinentes pour Qt 4.7.x.

Cet article est une traduction autorisée de **State of HTML5 <canvas> in QtWebKit**, par Andreas Kling.

N'hésitez pas à commenter cet article !
Commentez

I - L'article original....3
II - Introduction rapide....3
III - Conformité....3
IV - Performances....4
V - Les travaux futurs....4
VI - Démos....5
VII - Divers....5

Copyright ® 2010 Andreas Kling. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.
https://qt-labs.developpez.com/webkit/etat-html5-canvas/

I - L'article original

Les *Qt Labs Blogs* sont des blogs tenus par les développeurs de Qt, concernant les nouveautés ou les utilisations un peu extrêmes du framework.

Nokia, Qt et leurs logos sont des marques déposées de *Nokia Corporation* en Finlande et/ou dans les autres pays. Les autres marques déposées sont détenues par leurs propriétaires respectifs.

Cet article est la traduction du billet **State of HTML5 <canvas> in QtWebKit**, par Andreas Kling.

II - Introduction rapide

<canvas> est une nouvelle fonctionnalité dans HTML5 - un élément de bloc avec une API de dessin impératif. Il a été (un peu exagérément) introduit comme un "tueur de Flash". Vous pouvez l'utiliser pour créer des pages Web graphiquement riches et des démonstrations de nouvelles technologies voient le jour quotidiennement. La fonctionnalité provient de WebKit, développée initialement pour le tableau de bord de Mac OS X. Naturellement, QtWebKit la supporte.

« Qt! » joliment écrit en utilisant la démo FlowerPower.

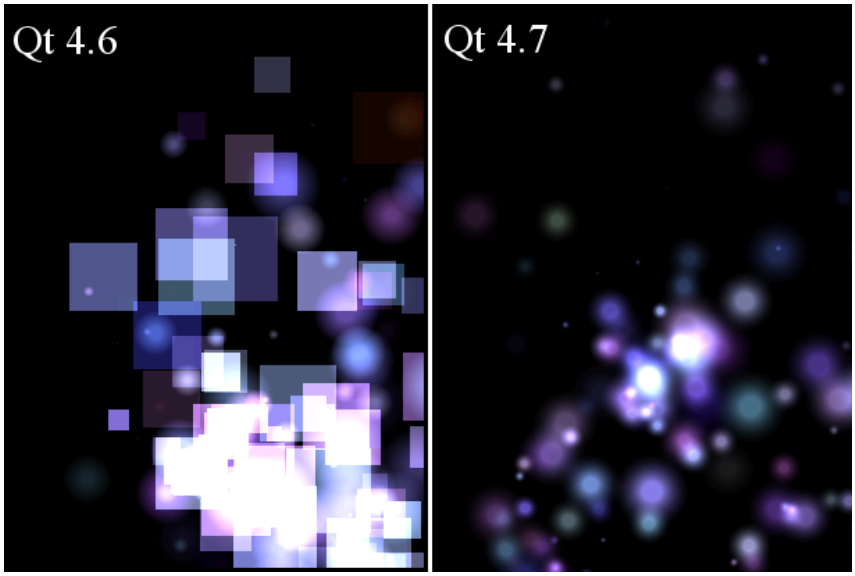
III - Conformité

La conformité des spécifications est élevée - nous passons plus de 90% de **la suite de test de Philip Taylor** et il existe des correctifs à l'étude pour de nombreux échecs restants. En outre, certains des tests sont devenus invalides après des changements dans les spécifications. La plupart des erreurs de rendu connues ont été rectifiées et nous nous approchons d'une correspondance parfaite avec Chromium et Safari.

La grande exception est le cas des dégradés radiaux ; HTML5 définit ses dégradés radiaux sur la base de l'implémentation originale de CoreGraphics qui diffère des dégradés SVG standards (et c'est ce que QRadialGradient implémente). J'ai essayé de convaincre les développeurs de **WHATWG** d'adopter les dégradés sur le style SVG pour les canevas, mais ils sentaient qu'il était trop tard pour un tel changement. Heureusement, je dois encore voir quelqu'un qui doit exploiter les possibilités supplémentaires des dégradés sur le style CoreGraphics "à l'état sauvage".

Copyright ® 2010 Andreas Kling. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.
https://qt-labs.developpez.com/webkit/etat-html5-canvas/

Nous avons récemment obtenu le support pour les ombres floues grâce à Ariya Hidayat. Il s'agissait de la dernière grande fonctionnalité qui manquait dans notre implémentation.



L'une des nombreuses corrections de bogues depuis Qt 4.6, des captures d'écran de la démo Parcycle.

IV - Performances

Les performances ont été en progression constante depuis la sortie de Qt 4.6 et dans Qt 4.7, vous verrez de grandes accélérations grâce aux efforts de **Benjamin Poulain** dans la vectorisation du code de mélange de QPainter. Nous ne sommes pas encore au niveau de Skia (boîte à outils de Chromium) dans le mélange de pixmaps lors d'agrandissements et/ou de transformations, mais Olivier Goffart travaille actuellement dessus et les chiffres préliminaires semblent bons.

Comme il n'existe pas d'objet "couleur" natif, les traits et les couleurs de remplissage sont gérés en utilisant des chaînes de caractères CSS. Auparavant, ils seraient passés par le parseur de CSS généré (lex) qui est très lent. J'ai récemment déposé un analyseur de couleurs `rgba()` optimisé qui aide à améliorer n'importe quelle application qui peint avec de nombreuses couleurs différentes.

La vitesse de manipulation des pixels est acceptable, pourtant nous n'avons pas encore atteint nos limites. Principalement parce que nous sommes obligés de faire la conversion RGB/BGR à la volée dans `putImageData()` et `getImageData()`. Ceci pourrait être atténué par l'ajout de `QImage::Format_ABGR32_Premultiplied` mais je pense que ce serait exagéré pour le moment. De façon plus réaliste, on pourrait au moins vectoriser la conversion de format à l'intérieur de WebKit.

Notre plus grande faiblesse est le traceur de chemin - le traceur et le rastériseur de chemin dans Qt sont lents par rapport aux autres boîtes à outils. Certaines personnes chez Google, travaillent sur des chemins accélérés par le GPU, ce que nous surveillons avec grand intérêt.

La seconde plus grande faiblesse est le rendu des dégradés. Qt met en cache les données de soixante dégradés au plus, mais il est très courant pour les applications à base de canevas d'utiliser beaucoup plus de dégradés uniques, ce qui rend le mécanisme de cache très coûteux. L'optimisation (et la vectorisation) de ces opérations est dans la file d'attente des projets de recherche.

V - Les travaux futurs

Les ingénieurs de Google travaillent sur une implémentation de canevas accélérée matériellement pour WebKit. Espérant que nous serons en mesure de profiter de ce travail sur les plates-formes où ça sera significatif.

Copyright ® 2010 Andreas Kling. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.
https://qt-labs.developpez.com/webkit/etat-html5-canvas/

Actuellement, nous regardons l'optimisation de notre code de filtrage bilinéaire qui excelle actuellement dans de nombreux profils de performance.

VI - Démos

Très bien, assez parlé. Voici quelques-unes de mes démos de <canvas> préférées pour le plaisir de vos yeux :

- **Parcycle**
- **Interferoplasma**
- **Monster**
- **Trail**
- **Canvas Cycle**

Des tonnes de démos supplémentaires sur **canvasdemos.com**

VII - Divers

Merci à **Buffering** ainsi qu'à **dourouc05** et **jacques_jean** pour leur relecture !

Copyright ® 2010 Andreas Kling. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.
https://qt-labs.developpez.com/webkit/etat-html5-canvas/