

Imprimer dans un repère en mm au lieu de pixels

par Jean-Luc Mellet(Alphomega) Traduit en C++Builder par Jean-Pierre
Blondelle(blondelle) Relecture par Alexandre Pottiez(pottiez) et par Cl@udius

Date de publication : 01/10/2007

Dernière mise à jour : 01/10/2007

Suite à question posée sur le forum nzn.fr.delphi, j'ai décidé de traduire ce petit article sur les calculs de position en millimètres sur le canvas des imprimantes.

Il faut savoir que le canvas de l'objet TPrinter n'accepte (comme tous les TCanvas) que des coordonnées en pixels. L'être humain n'étant pas habitué à ce genre de mesure, il est difficile pour lui de se repérer dans une telle dimension. Le but de la man?uvre est donc d'utiliser quelques petits calculs simples pour se repérer dans un système plus conventionnel (au sens humain de la chose).
Pour réaliser cela, il va nous falloir des éléments de base. On a alors 2 solutions.

L'objet Printer

L'objet Printer est automatiquement créé par C++ Builder dès que le code le réclame. Il est nécessaire d'inclure dans l'entête " #include <Printers.hpp> ". Il n'est donc nul besoin de le créer explicitement. On pourra faire par exemple:

```
// ce qui nous donnera la largeur imprimable de la page pour l'imprimante en cours
LargeurCanvas = Printer()->PageWidth;
```

Le seul problème est que, pour obtenir un résultat valable, il faut obligatoirement lancer une impression avec la procédure BeginDoc. Cela implique donc le démarrage d'une impression, l'obtention des valeurs, puis l'abandon de la procédure. Vous me direz que si on veut imprimer, on est bien obligé d'en passer par là. Mais on peut avoir besoin de connaître les dimensions de la page imprimable sans pour autant lancer une impression. Un prochain article sur "Comment réaliser un aperçu avant impression" montrera pourquoi. La procédure normale serait donc:

```
Printer()->BeginDoc();
LargeurCanvas = Printer()->PageWidth;
Printer()->Abort(); // on recupere les valeurs puis on stop l'impression
```

L'autre solution consiste à utiliser l'API GetDeviceCaps qui nous fournit tous les renseignements nécessaires sans lancer d'impression.

L'API GetDeviceCaps

Cette fonction de Windows nécessite un handle et une constante pour indiquer quel type de résultat on veut obtenir. Pour l'objet TPrinter, le handle sera Printer()->Canvas->Handle. Dans notre cas, nous utiliserons les constantes LOGPIXELSX et LOGPIXELSY qui permettent à GetDeviceCaps de renvoyer respectivement le nombre de pixels par pouce logique. On aura noté qu'il existe une constante pour les valeurs en X et une pour les valeurs en Y. En ce qui concerne l'écran, les 2 valeurs seront différentes, car physiquement, un pixel est plus haut que large. Sur une imprimante, les 2 valeurs seront (généralement) identiques, car dans ce cas, un pixel représente un point parfaitement rond. Aussi, l'une et l'autre constante renverront le même résultat.

Pour obtenir le nombre de pixels par pouce pour l'imprimante, nous écrirons donc:

```
Int XPixelsParPouce = GetDeviceCaps(Printer()->Handle, LOGPIXELSX);
Int YPixelsParPouce = GetDeviceCaps(Printer()->Handle, LOGPIXELSY);
```

Ainsi, pour commencer l'impression d'un texte à 1 pouce du bord gauche de la feuille, et à 1 pouce du haut de la feuille, on écrira:

```
Printer()->Canvas->TextOutA(XPixelsParPouce, YPixelsParPouce, "Mon texte");
```

Maintenant, si nous voulons travailler en millimètres pour plus de commodités, nous devons transformer nos coordonnées en millimètres. Comme vous le savez certainement, 1 pouce = 25.4 mm, donc 1 mm = 1 / 25.4 pouce. Nous pouvons donc calculer directement:

Puisque nous savons maintenant qu'un pouce réel équivaut à un certain nombre de pixels (XPixelsParPouce), nous pouvons facilement déterminer la valeur d'un pixel en millimètres.

1 pouce = XPixelsParPouce = 25.4 mm ==> 1 pixel = 25.4 / XPixelsParPouce.

A partir de là, on peut écrire 2 fonctions simples pour transformer les millimètres que nous connaissons en pixels reconnus par notre imprimante:

Ainsi, pour imprimer un texte à 50 millimètres du bord gauche de la feuille et 120 millimètres du bord haut, on écrira:

```
int Marge_Gauche = 50;
int Marge_Haute = 120;
int Millimetres2PixelsX = Marge_Gauche / (25.4 / XPixelsParPouce);
int Millimetres2PixelsY = Marge_Haute / (25.4 / YPixelsParPouce);
Printer()->Canvas->TextOutA( Millimetres2PixelsX,
Millimetres2PixelsY, "Mon Texte à (50, 120)");
```

ATTENTION: Ce code imprimera le texte avec un léger décalage supplémentaire. Il s'agit de la zone non imprimable définie par l'imprimante elle-même. Pour connaître la taille de cette zone non imprimable, vous utiliserez les constantes PHYSICALOFFSETX et PHYSICALOFFSETY. Vous obtiendrez alors les valeurs en pixels dont vous devrez tenir compte pour positionner vos éléments à imprimer.

```
int NomImprimableHorizPixelsX = GetDeviceCaps(Printer()->Handle, PHYSICALOFFSETX);
int NomImprimableVertiPixelsY = GetDeviceCaps(Printer()->Handle, PHYSICALOFFSETY);
```

Nous pouvons alors terminer ainsi pour imprimer notre texte à exactement 50 mm du bord gauche et 120 mm du bord haut:

```
int Marge_Gauche = 50;
int Marge_Haute = 120;
Printer()->BeginDoc(); // on obtient le nombre de pixels par pouce pour l'imprimante
int XPixelsParPouce = GetDeviceCaps(Printer()->Handle, LOGPIXELSX);
int YPixelsParPouce = GetDeviceCaps(Printer()->Handle, LOGPIXELSY);
int Millimetres2PixelsX = Marge_Gauche / (25.4 / XPixelsParPouce);
int Millimetres2PixelsY = Marge_Haute / (25.4 / YPixelsParPouce);
int NomImprimableHorizPixelsX = GetDeviceCaps(Printer()->Handle, PHYSICALOFFSETX);
int NomImprimableVertiPixelsY = GetDeviceCaps(Printer()->Handle, PHYSICALOFFSETY);
// on imprime notre texte à exactement 50 mm du bord gauche
Printer()->Canvas->TextOutA(Millimetres2PixelsX - NomImprimableHorizPixelsX,
Millimetres2PixelsY - NomImprimableVertiPixelsY, "Mon Texte à (50, 120)");
Printer()->EndDoc();
```

Voilà! Vous avez maintenant les éléments de base pour imprimer ce que voulez, où vous voulez, dans un repère plus habituel.

Bonne continuation et à bientôt!