

Créer des applications Tablet PC en .Net

par Louis-Guillaume MORAND ([Page perso de Louis-Guillaume MORAND](#))

Date de publication : 11/01/2008

Dernière mise à jour :

Petit article de présentation pour créer des applications tirant partie des fonctionnalités d'un Tablet PC

Introduction.....	3
1 - Présentation.....	4
1.1 - Préparation.....	4
1.2 - Présentation des composants TabletPC.....	5
2 - Développer une application utilisant le stylet Tablet PC.....	6
2.1 - Dessiner.....	6
2.2 - Gommer.....	7
2.3 - Sélectionner.....	7
3 - Utiliser la reconnaissance d'écriture Tablet PC.....	9
4 - Implémenter la gestion de la reconnaissance de mouvements.....	10
5 - Pour aller plus loin avec les classes Tablet PC.....	11
Conclusion.....	12
Téléchargements et remerciements.....	13

Introduction

Depuis des années, les périphériques Tablet PC se développent et ne se limitent plus à des gadgets pour personnes mobiles et riches. Alors que l'utilisation des ordinateurs évoluent (tablettes graphiques, écrans tactiles, etc), les interfaces des logiciels de tous les jours évoluent également.



1 - Présentation

Le développement TabletPC pourrait paraître au premier abord comme un développement spécifique et qu'il convient de maîtriser, comme l'est le développement mobile. C'est tout le contraire, le développement TabletPC est très simple et consiste à rajouter des fonctionnalités de dessins au stylet et éventuellement de reconnaissance d'écriture, rien de plus.

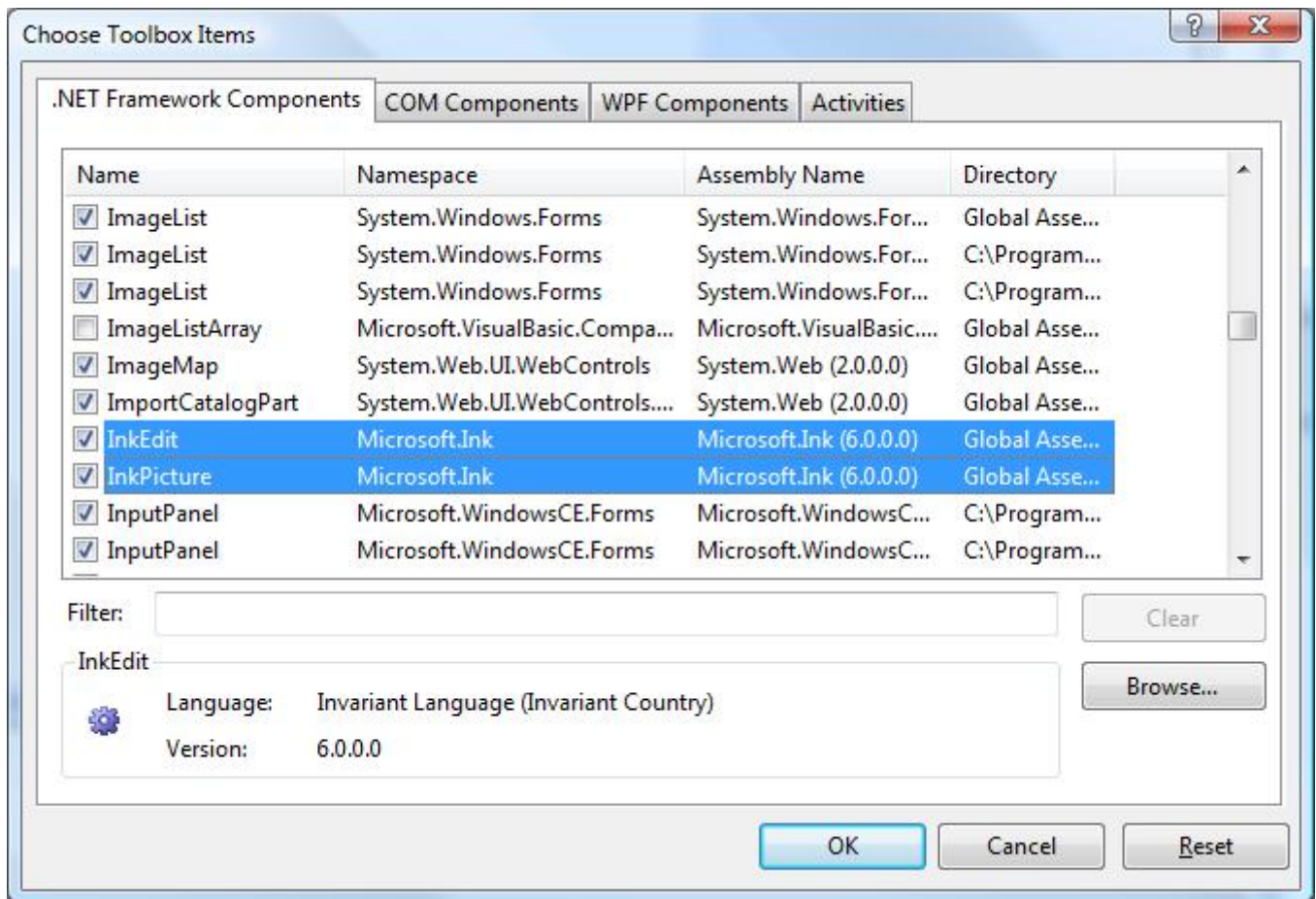
Oui, mais si on a pas de TabletPC, cet article ne nous intéresse pas. **Erreur!!! Nous allons ici voir comment créer des applications optimisées pour TabletPC mais également utilisables sans TabletPC.** Je vous assure que dans de nombreux cas, ceci pourra vous être utile, car je l'ai moins même rencontré pour un besoin métier alors que l'environnement périphérique de l'application n'utilise aucun périphérique de capture de type TabletPC.

1.1 - Préparation

Gérer un périphérique externe, capter ses messages, et agir en conséquence demande énormément de boulot et ne se fait pas en quelques lignes de code. Heureusement pour nous, Microsoft a mis à notre disposition différentes assemblées permettant de passer outre cette préparation, et d'utiliser des classes et méthodes toutes prêtes vous permettant d'être directement fonctionnels.

Il n'existe pas une assembly mais plusieurs assemblées utilisables mais toutes devraient vous permettre d'arriver au résultat escompté. Selon que vous utilisiez des SDK (AxInterop.INKEDLib.dll), que vous utilisiez la librairie installée dans le répertoire system32 (INKEDLib.dll) ou que vous téléchargiez sur  **la MSDN la dernière version** de l'assembly (Microsoft.Ink.dll), vous aurez des composants ayant sensiblement le même nom et les mêmes propriétés. Néanmoins, parce que depuis .Net, nous aimons utiliser des classes dites "wrapper", cachant la mécanique inutile qui se trouve au fin fond d'objets COM, nous utiliserons dans cette article, la  **version 6.0 de Microsoft.Ink.**

Une fois l'assembly téléchargée sur votre ordinateur, vous allez ajouter les contrôles qu'elle contient, dans la barre de composants de Visual Studio afin de pouvoir les déposer sur les fenêtres de vos applications. Ouvrez Visual Studio, passez en mode design d'une quelconque fenêtre et sur le panneau de composants, faites bouton droit > Choisir éléments... puis parcourez votre disque dur pour choisir la dll que nous venons de télécharger juste avant, et cochez les deux contrôles qu'elle contient (voir capture ci-dessous).



Il vous restera enfin, à rajouter le using nécessaire pour pouvoir appeler les classes depuis votre programme.

```
using Microsoft.Ink;
```

1.2 - Présentation des composants TabletPC

Il existe plusieurs classes utiles dans Microsoft.Ink mais seuls deux véritables composants pourront vous servir à améliorer vos interfaces d'application.

Nous avons tout d'abord InkEdit, contrôle héritant de RichTextBox et permettant d'avoir une textbox gérant le texte riche (RTF) mais également l'entrée du stylet. Plus encore, comme nous le verrons dans un chapitre futur, ce contrôle permet de gérer la reconnaissance d'écriture de façon extrêmement simple, j'insiste sur le "extrêmement". Puis, nous avons le contrôle InkPicture, qui hérite de PictureBox, et permet lui aussi de gérer l'entrée du stylet pour dessiner directement sur le composant.

Dans les chapitres suivants, nous allons voir dans quels cas et surtout comment utiliser ces composants, et comment ils peuvent vous aider pour donner des fonctionnalités spécifiques à vos applications .Net.

2 - Développer une application utilisant le stylet Tablet PC

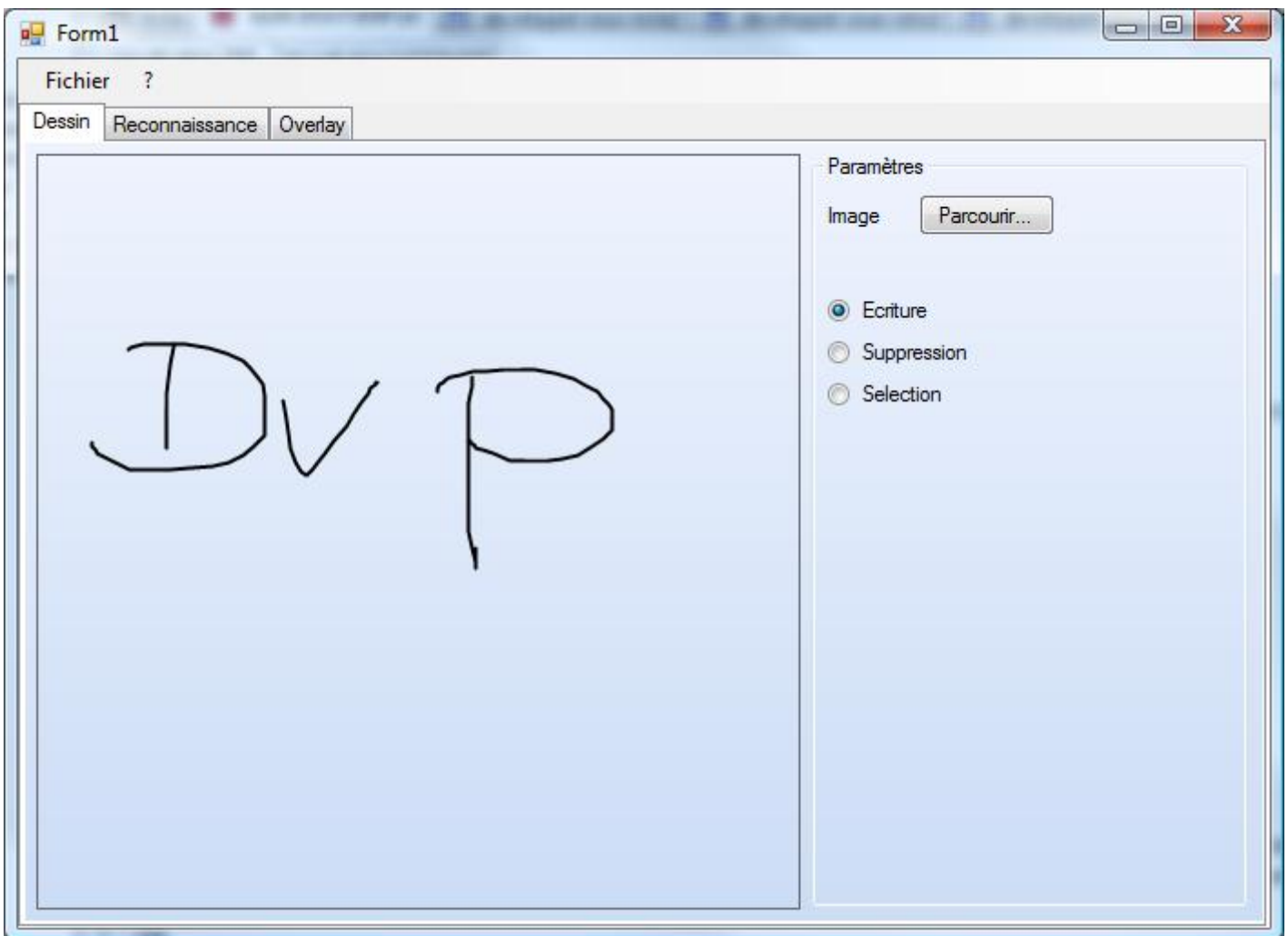
Pour ceux qui ont déjà utilisé un TabletPC, quelle est selon vous la première chose que l'on est content de faire, outre la sélection des menus via le stylet? Tout simplement, l'annotation. L'annotation consiste à prendre un document ou une image (un plan par exemple) et d'y ajouter des remarques. Supposons que vous ayez une carte de quartier et que vous souhaitiez ajouter des points de repère très facilement. Vous pouvez alors ouvrir l'image avec un logiciel de retouche d'image comme MSPaint ou alors, utiliser le TabletPc.

Comment feriez-vous en tant normal? Vous pourriez utiliser un composant de type pictureBox, mettre l'image en fond d'écran et vous servir des différentes classes et méthodes de System.Drawing pour, selon les mouvements de la souris, dessiner sur l'image. Ceci n'est pas spécialement long mais pourrait être fait de façon beaucoup plus facile: en utilisant les composants TabletPc.

2.1 - Dessiner

Puisque la solution de System.Drawing est "complexe" à mettre en oeuvre, nous allons utiliser un composant dédié à cette tâche: le contrôle InkPicture.

Pour l'utiliser, vous n'avez qu'à le déposer sur votre formulaire et...c'est tout. Oui, pas la moindre manipulation ou la moindre ligne de code supplémentaire n'est nécessaire. Une fois le programme lancé, vous pouvez dessiner comme le montre la capture suivante.



Puisque le composant InkPicture hérite de PictureBox, il est possible de mettre une image en arrière plan, de dessiner dessus puis de sauvegarder le résultat.

```
inkPicture1.Image.Save("mon image .jpg", ImageFormat.Jpeg);
```

2.2 - Gommer

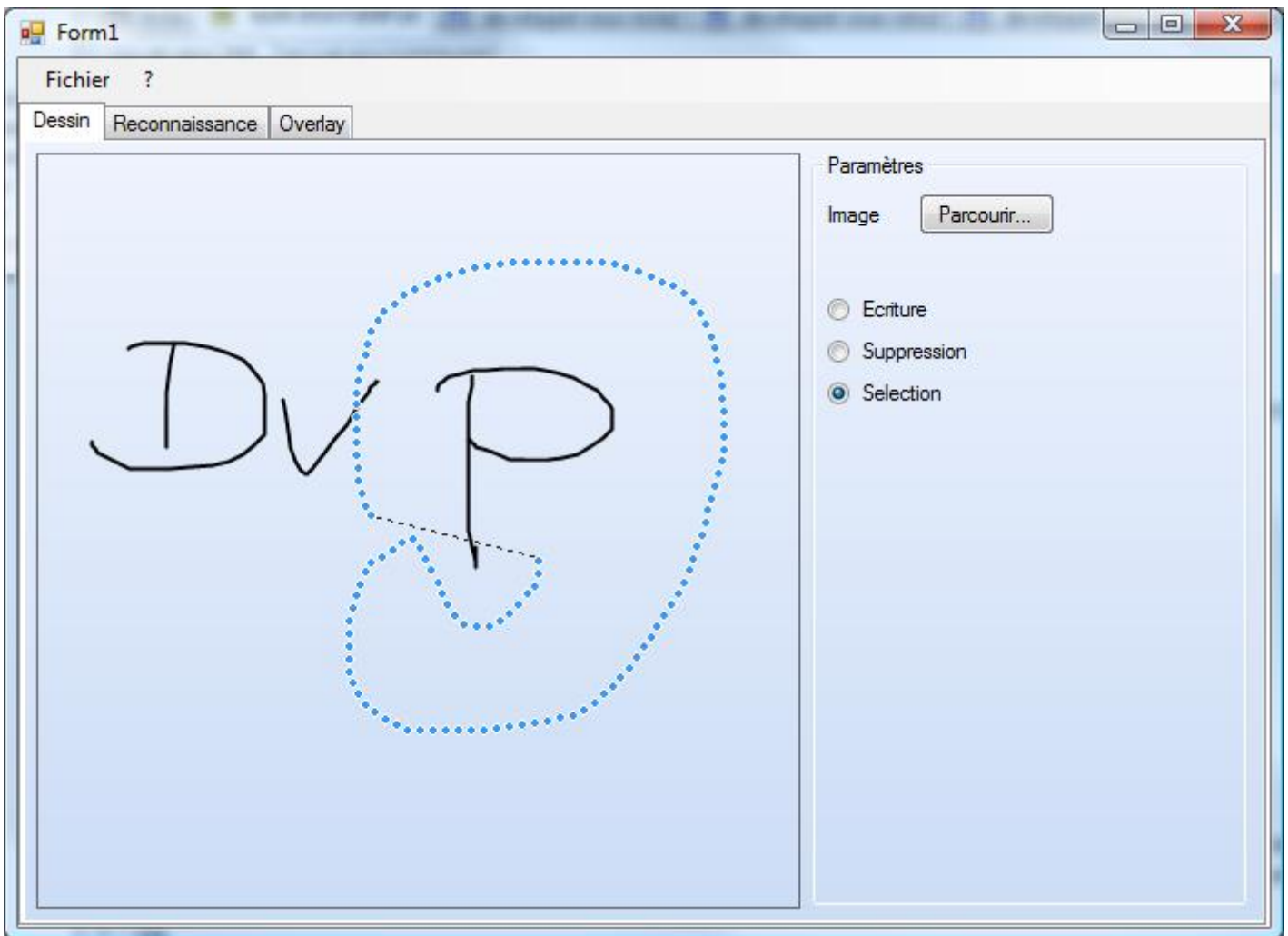
Une chose intéressante avec le dessin via TabletPC, est que vous pouvez effacer les "traits" que vous avez faits. Il ne s'agit pas de gommer comme vous feriez avec une gomme normale, mais plutôt d'une gomme intelligente qui sélectionne un trait et le gomme entièrement avec un seul clic. Cette gomme fait automatiquement la différence entre les différents traits (même si ces derniers se croisent) et ne supprime que celui que vous avez désigné. Si cela ne vous semble pas clair, changer le mode d'édition de votre InkPicture et cliquez sur l'un des traits tracés

Pour activer la suppression, il suffit de changer le mode d'édition de votre composant:

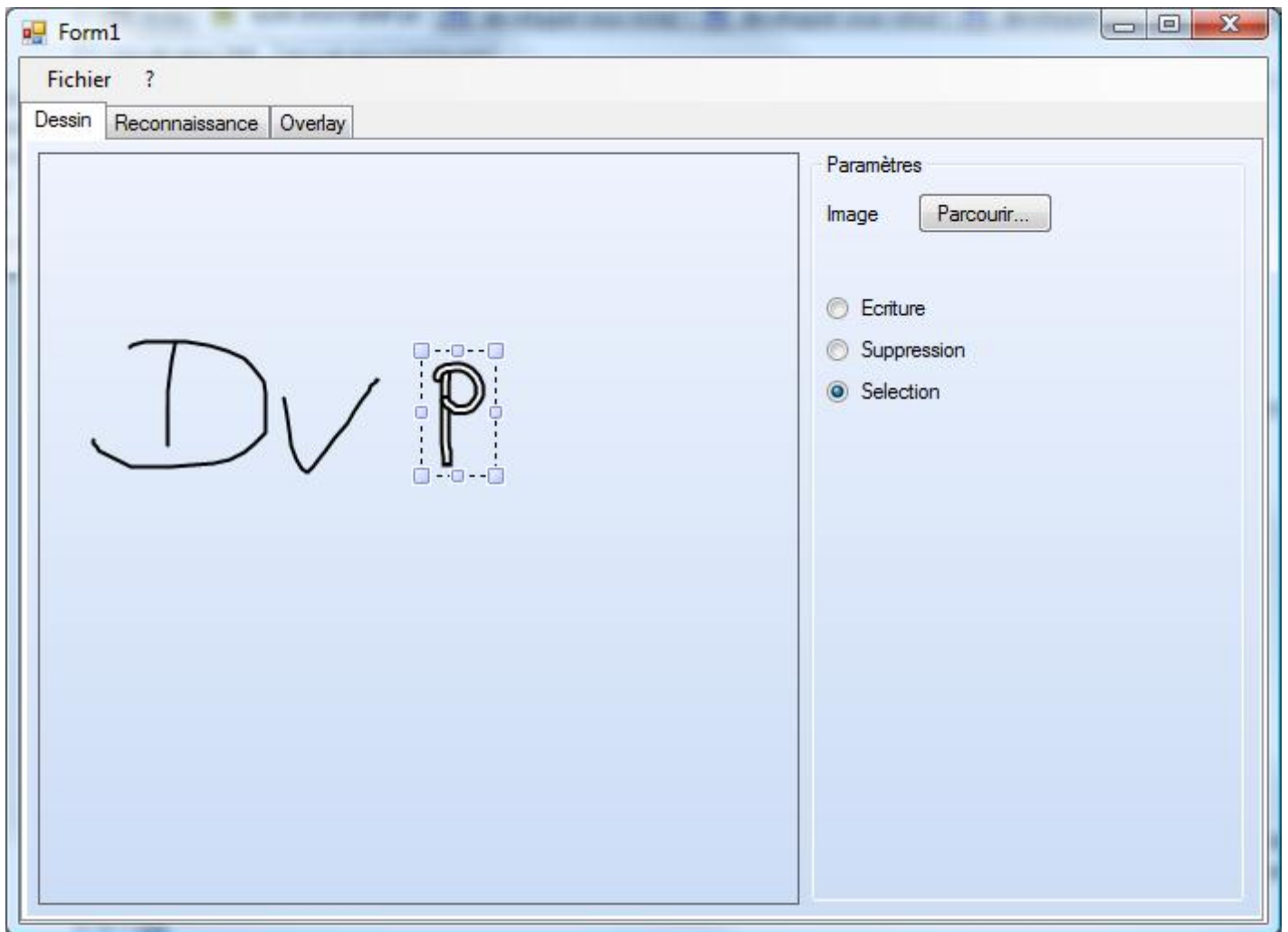
```
inkPicture1.EditingMode = InkOverlayEditingMode.Delete;
```

2.3 - Sélectionner

Comme la suppression, il est également possible de passer en mode **Selection** sur votre contrôle, c'est-à-dire pouvoir sélectionner les "traits" dessinés, soit en les entourant d'une forme de sélection, soit en cliquant une fois dessus:



Soit pour pouvoir modifier la taille et la position du trait sélectionné



Pour activer le mode **Selection**, une seule ligne de code:

```
inkPicture1.EditingMode = InkOverlayEditingMode.Select;
```

3 - Utiliser la reconnaissance d'écriture Tablet PC

Dessiner sur des images et les sauvegarder est une chose intéressante mais il est surtout possible de faire bien plus que cela. Il est possible notamment d'utiliser la reconnaissance d'écriture. Si vous avez un Palm, ou un SmartPhone marchant avec un stylet, ou alors si vous avez une Nintendo DS, vous avez déjà sûrement rencontré une application où vous écriviez "manuellement" les lettres et elles étaient reconnues plus ou moins correctement.

Comment pourrions nous aussi développer une application permettant l'utilisation à la fois du clavier et du stylet? Plus encore, comment permettre l'utilisation du stylet...sans stylet, comme par exemple, une fonctionnalité d'accessibilité pour une personne ayant du mal à utiliser un clavier? La réponse est: de façon très simple.

Nous avons précédemment utilisé le contrôle InkPicture, et donc en toute logique, c'est le contrôle InkEdit que nous allons maintenant utiliser.

Celui-ci est non seulement très simple d'utilisation, avec seulement quelques propriétés, mais surtout, il fait tout tout seul. Placez un contrôle InkEdit sur votre formulaire et compilez:

Vous devriez alors voir apparaître une RichTextBox dans laquelle vous êtes capable d'écrire mais aussi de dessiner.




*Si vous n'avez pas de Tablet PC, sous la main, réglez la propriété **UseMouseForInput** sur **true**. Cela permettra de simuler le stylet à l'aide de votre souris*

Ecrivez alors votre nom à l'aide de votre stylet (ou souris) et puis attendez 2 secondes. Le contrôle tentera alors de reconnaître votre écriture et ajoutera le mot à la suite du texte déjà présent dans la TextBox. Pas besoin de lui apprendre votre écriture, celui-ci est assez malin pour trouver correctement le mot dans la plupart des cas.

Vous pouvez essayer des phrases entières, ou avec des chiffres, tout est pris en compte.

Il est bien entendu possible de paramétrer différentes choses. Vous avez tout d'abord le timeout de reconnaissance qui peut être réglé via la propriété **RecoTimeout**. Si la valeur est mise sur 0 (zéro) alors la reconnaissance automatique est désactivée.

Vous avez également la propriété FACTOID qui permet de filtrer et de guider la reconnaissance du contrôle selon le contexte. Vous pourriez par exemple, utiliser un factoid **PostalCode**, ou **Email**, ou encore **Telephone**.

Enfin, pour parfaire la reconnaissance de votre contrôle, vous pouvez personnaliser l'objet  **Recognizer** de votre InkEdit.

4 - Implémenter la gestion de la reconnaissance de mouvements

Imaginons maintenant que vous vouliez aller plus loin avec le stylet. Cela ne serait-il pas merveilleux si vous pouviez contrôler des actions de votre application en fonction du mouvement dessiné par le stylet? Et bien, c'est ce que nous allons mettre en place dans ce chapitre.

Une fois encore, il suffira de très peu de lignes de code puisque les composants Tablet PC sont faits pour fonctionner seuls. Placez donc un contrôle InkPicture sur votre application, puis réglez sa propriété **CollectionMode** sur **GestureOnly**. Ensuite, dans le code, au chargement du formulaire, vous allez devoir activer l'écoute des mouvements.

Pour cela, vous devez utiliser la méthode SetGestureStatus et préciser les mouvements que vous voulez écouter.

```
inkPicture2.SetGestureStatus(ApplicationGesture.AllGestures,true);
```

Vous devez pour finir vous abonner à l'évènement Gesture de votre composant et obtenir une méthode ayant pour squelette le code suivant:


```
private void inkPicture2_Gesture(object sender, InkCollectorGestureEventArgs e)
{
    //code
}
```

Cet évènement est déclenché à chaque fois qu'un mouvement est reconnu. Il faut savoir que plusieurs mouvements sont reconnus au même moment mais que ceux-ci sont classés par ordre de fiabilité. Ainsi, si l'application hésite entre un cercle et un carré, le premier geste envoyé définira la plus grande probabilité d'avoir détecté le bon geste. Nous souhaitons donc nous intéresser qu'au geste le plus fiable.

```
Gesture G = e.Gestures[0];
```

Nous pouvons ensuite, définir un minimum de fiabilité attendu pour ce geste. Supposons que le geste Rond ferme l'application, vous aimeriez être sûr que la probabilité que ce soit un rond soit au moins de 80%. Vous pouvez pour cela vous baser sur l'indice de confiance du geste, via la propriété **Confidence**. Il ne vous reste alors qu'à traiter et agir en fonction du geste reconnu:



```
if (G.Confidence == RecognitionConfidence.Intermediate || G.Confidence == RecognitionConfidence.Strong)
{
    switch (G.Id)
    {
        case ApplicationGesture.Circle:
            lblGesture.Text = "Vous avez dessiné un cercle";
            break;
        case ApplicationGesture.Triangle:
            lblGesture.Text = "Vous avez dessiné un triangle";
            break;
        case ApplicationGesture.Scratchout:
            lblGesture.Text = "Vous avez dessiné un gribouilli";
            break;
        case ApplicationGesture.Down:
            lblGesture.Text = "Vous avez dessiné vers le bas";
            break;
    }
}
```

Vous noterez que les gestes reconnaissables sont prédéfinis dans une énumération donc la liste complète et la description peut être trouvée  [ici](#).

Et voilà, il est maintenant très facile et très "in" de laisser votre application être contrôlée par des mouvements de stylet ou de souris, donnant l'impression à l'utilisateur de pouvoir agir avec la main comme le seront sûrement les applications de demain.

5 - Pour aller plus loin avec les classes Tablet PC

Si l'utilisation des composants InkPicture et InkEdit vous permettront d'ajouter de nombreuses fonctionnalités à vos applications, il peut être utile de creuser un peu plus dans le namespace Microsoft.Ink afin d'y voir si l'on pourrait trouver des classes pouvant nous être utile.

Je laisse le soin de lire la  **documentation officielle de Microsoft.Ink** ou bien d'utiliser Reflector sur l'assembly pour y trouver les éléments disponibles et je me contenterai de ne vous en présenter qu'un, qu'il m'a été donné d'utiliser: il s'agit de la classe  **InkOverlay**. Cet objet est utile lorsque "la reconnaissance de l'écriture ne vous intéresse pas mais qu'au contraire, vous êtes intéressés par la couleur, la taille, la forme et/ou la position de l'encre". Plus encore, InkOverlay, vous permet de dessiner sur un autre composant que ceux provenant du namespace Microsoft.Ink, comme un panel par exemple.

Créez votre objet InkOverlay puis dans son constructeur, définissez le contrôle sur lequel il dessinera:

```
InkOverlay _inkOverlay = new InkOverlay(pnlDessin.Handle);
```

Activez l'objet. Si vous ne le faites pas, rien ne sera pris en compte.

```
_inkOverlay.Enabled = true;
```

Une fois ceci fait, vous pouvez dessiner sur votre panel, comme vous le feriez sur un contrôle InkPicture. Mais vous allez aussi pouvoir tout personnaliser. Voici comment régler la couleur et la taille du "pinceau":

```
_inkOverlay.DefaultDrawingAttributes.Color = Color.Red;  
_inkOverlay.DefaultDrawingAttributes.Width = 50;
```


InkOverlay est fort utile car tout ce que fait InkPicture, il le fait mais bien plus encore comme la détection des périphériques Tablet PC, les événements divers, etc.

Conclusion

Le développement pour Tablet PC se révèle très simple car il ne fait qu'ajouter la gestion du stylet pour contrôler les éléments, et d'utiliser la reconnaissance d'écriture afin de rendre les applications plus user-friendly. Si un jour, vous êtes amené à développer une application Winform, posez-vous simplement la question de savoir si vous pourriez faciliter son utilisation, en implémentant rapidement les fonctionnalités TabletPC. Si jamais l'application n'est utilisée que sur un ordinateur ordinaire, vous n'aurez alors perdu que quelques lignes de code mais votre conscience d'avoir fait au mieux en ressortira satisfaite.

Téléchargements et remerciements

Pour les sources de l'article, cliquez [ici](#)

Petit  [lien](#) vers la dll qui pourrait servir à certains.

Je tiens à remercier [Debug](#) pour ses conseils et corrections avisés sur l'article.