Bing Maps for Enterprise – Intégration dans une application WPF et pour la table Microsoft Surface

Sommaire

Rappel des derniers événements 4

Fin de vie du plugin 3D de la plateforme Bing Maps 4

Nouvelle version de la table Surface : Surface 2.0 5

Résumé des possibilités d'intégration 6

Présentation détaillée des solutions techniques 7

Utilisation du WebBrowser et de l'API Bing Maps AJAX v7 7

Utilisation du WebBrowser et du contrôle Silverlight de Bing Maps 8

Utilisation d'un navigateur alternatif et de l'API Bing Maps AJAX V7 9

Utilisation d'un navigateur alternatif et du contrôle Silverlight 10

Utilisation d'une application Silverlight directement en WPF 11

Utilisation du plugin 3d à travers le contrôle InfoStrat.VE – CodePlex 12

Utilisation du contrôle GreatMaps for WPF – CodePlex 13

Utilisation d'un contrôle commercial dédié – Telerik RadMap 14

Développement personnalisé natif WPF 15

Cas particuliers de la table Microsoft Surface 16

La table Surface en version 1 (Microsoft Surface Table V1) 16

La table Surface en version 2 (Microsoft Surface Table V2) 17

Principales différences 18

Aspect logiciel : 19

Considérations pour vos applications pour la table Surface V1.0 20

Evolution des développements 20

Conclusions 21

Autres pistes à explorer : 21

A travers ce précédent article dédié à la réalisation d'application de cartographie à travers une application WinForm et/ou WPF (Windows Presentation Foundation), nous avions déjà abordé certaines techniques d'implémentation en pondérant chacune de ces solutions en fonction du contexte.

Cet article a pour but d'aider les entreprises souhaitant effectuer la migration de leurs applications existantes sur la table Surface (basée sur l'application Concierge, le composant InfoStrat.VE ou autres…) ou tout simplement intégrer de la cartographie au sein d'application WPF ou WinForm.

Au sein de Wygwam, nous avons régulièrement des demandes pour réaliser ce type d'application ou de contrôle d'intégration permettant d'apporter des possibilités cartographiques (affichage de carte, utilisation des contenus) aux applicatifs utilisant la technologie WPF.

C'est donc avec mon collègue Ludovic Czarnecki que j'ai pu tester les différentes solutions techniques présentées ci-dessous dans un contexte d'exécution de la table Microsoft Surface V1.

N'hésitez donc pas à nous contacter si vous avez ces besoins d'intégration ou si vous souhaitez davantage d'informations concernant une technique en particulier.

Rappel des derniers événements

Afin de comprendre la raison de cette mise à jour concernant les solutions d'intégration disponibles, il faut parcourir à nouveau l'actualité des derniers mois des technologies concernées.

Fin de vie du plugin 3D de la plateforme Bing Maps

Microsoft a annoncé en novembre 2010 la fin de vie programmée en novembre 2011 du plugin 3d disponible depuis plusieurs années et surtout utilisable à travers l'API en version 6.3.

Ce plugin 3d est pourtant utilisé dans de nombreux cas pour intégrer cette plateforme au sein d'application WPF et plus particulièrement sur la table Surface V1.

En effet, l'application Concierge mais également le contrôle Infostrat.VE disponible sur CodePlex utilisent ce plugin pour effectuer l'affichage.
Dès lors, même si ce plugin est installé (ou les assemblies nécessaires conservées), rien ne garantit son fonctionnement après la date indiquée notamment pour la récupération du modèle de terrain (DEM : Digital Elevation Model) ou pour les géométries 3d des bâtiments ou lieux connus.

Bref, le taux de pénétration réduit de ce plugin sur le poste grand public et l'évolution technologique ont motivé Microsoft a stoppé l'investissement sur ce dernier et à se concentrer sur d'autres possibilités de présentation et d'exploitation de ces informations et contenus 3D.

Nouvelle version de la table Surface : Surface 2.0

L'arrivée prochaine de la table Surface 2.0 avec une technologie différente (voir plus bas) pour analyser les contacts utilisateurs amène son lot de changement pour les développements réalisés jusqu'à présent.


La technologie logicielle implémentée modifie également les développements sur la table Surface, l'ensemble du SDK et du Framework dédié à ce produit en v2 a été modifié, on n'utilise par exemple plus d'événement de type Contact (spécifique à la table Surface v1) mais au contraire des événements de types Touch standard à l'ensemble des applications WPF dédiées au tactile.

Résumé des possibilités d'intégration

D'une manière générale, ces possibilités d'intégration concernent les applications WPF et WinForm. Les cas spécifiques associés à la table Microsoft Surface sont abordés plus bas.

Voici, avant de nous intéresser aux détails, les différentes techniques d'implémentation listées :

- Utilisation du WebBrowser et de l'API Bing Maps AJAX v7
- Utilisation du WebBrowser et du contrôle Silverlight de Bing Maps
- Utilisation d'un navigateur alternatif et de l'API Bing Maps AJAX V7
- Utilisation d'un navigateur alternatif et du contrôle Silverlight
- Utilisation d'une application Silverlight directement en WPF
- Utilisation du plugin 3d à travers le contrôle InfoStrat.VE – CodePlex
- Utilisation du contrôle GreatMaps for WPF – CodePlex
- Utilisation d'un contrôle commercial dédié
    - Contrôle Telerik – RadMap
- Développement personnalisé natif WPF

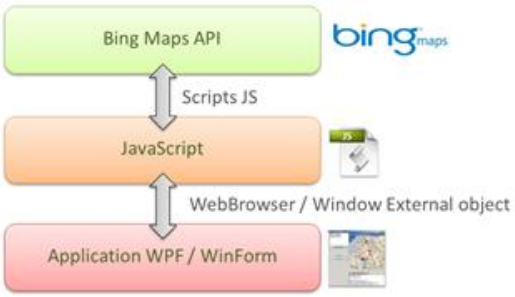
Présentation détaillée des solutions techniques

Après avoir résumé les solutions, nous allons nous intéresser aux détails d'implémentation mais également aux avantages et inconvénients de chacun d'elles.

Utilisation du WebBrowser et de l'API Bing Maps AJAX v7

Description technique :

Cette solution consiste à intégrer une instance du contrôle WebBrowser qui charge une page HTML distante ou un contenu HTML disponible en ressource intégrée à l'application.



Il suffit alors ensuite d'utiliser les méthodes **InvokeScript()** sur le contrôle **WebBrowser** (ou **InvokeScript()** sur la propriété **Document** du **WebBrowser** en WinForm) pour déclencher du code JavaScript depuis l'application .Net.

Du côté client, en JavaScript, il suffit d'utiliser l'élément **window.external** pour obtenir une référence vers l'objet passé dans la propriété **ObjectForScripting** du contrôle **WebBrowser** permettant ainsi de pouvoir appeler les méthodes de l'application .Net depuis le code client.

Analyse :

Avantages	**Inconvénients**
Méthode supportée	Intégration graphique limitée
Evolution aisée des développements et migration simple	Complexité de développement
Intégration simple du modèle de licence de la plateforme	Difficultés possibles lors d'un déploiement multi-systèmes

Mon avis :

Cette solution permets d'intégrer une application cartographie utilisant le contrôle interactif Bing Maps via l'API AJAX v7 en utilisant des techniques connues pour le développement autour de l'API.

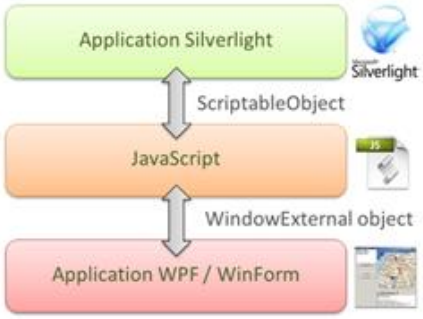
Le principal effort consiste à effectuer l'ensemble du code d'interopérabilité entre le WebBrowser/JavaScript et le code .Net.

Utilisation du WebBrowser et du contrôle Silverlight de Bing Maps

Description technique :

En utilisant la même technique que la précédente, on charge cette fois une page contenant le contrôle Silverlight de la plateforme Bing Maps.

Voici un vue d'ensemble de ce qui est opéré :



Il suffit donc de réalise une application .Net exposée en COM et établi comme propriété **ObjectForScripting** sur le composant **WebBrowser**.

Le code JavaScript assure la communication entre l'application WPF et le contrôle Silverlight.

Le contrôle Silverlight expose ses méthodes comme étant des **ScriptableMember** (en ajoutant l'attribut) et référence l'application comme ScriptableObject via la méthode **RegisterScriptableObject()** du contrôle WebBrowser en permettant ainsi d'accéder à ces différents attributs et méthodes.

Analyse :

Avantages	**Inconvénients**
Méthode supportée	Intégration graphique limitée
Performances du contrôle Silverlight	Complexité important du développement
Possibilité d'intégrer la logique applicative dans l'application Silverlight	Dépendance au plugin Silverlight

Mon avis :

Cette solution relativement proche de la précédente approche possède toujours les mêmes limites en termes d'intégration graphique mais permet d'exploiter les performances du contrôle Silverlight de Bing Maps for Enterprise.

La contrainte supplémentaire du plugin Silverlight augmente la complexité du développement des couches d'interopérabilités (voir Article en 4 parties : Bing Maps for Enterprise – Integration WPF : interopérabilité avec Silverlight) et nécessite une connaissance pointue des possibilités d'interopérabilité de chaque brique applicative.

Utilisation d'un navigateur alternatif et de l'API Bing Maps AJAX V7

Description technique :

A travers cette réalisation, on souhaite utiliser le contrôle basé sur Chromium afin de charger la page sous le même mode que le WebBrowser standard.

Avec ce contrôle, on observe la possibilité d'intégrer graphiquement le navigateur dans l'application WPF et d'opérer des manipulations impossibles à réaliser avec le contrôle WebBrowser standard : rotation, modification des dimensions, transformations avancées…

Pour simplifier l'utilisation de Chromium au sein d'une application WPF, il est possible d'utiliser le projet WPFChromium disponible via ces liens :

- Pour la version .Net 3.0 et .Net 3.5 : http://wpfchromium.codeplex.com/
- Pour la version .Net 4.0 : http://wpfchromium4.codeplex.com/

Analyse :

Avantages	**Inconvénients**
Intégration graphique facilitée	Méthode non-supportée
Utilisation du contrôle AJAX	Complexité de développement et limitations d'interopérabilité possibles
Intégration simple du modèle de licence de la plateforme	Quid de l'évolution

Mon avis :

Cette solution permets de s'affranchir de la limite d'intégration graphique, en effet avec le contrôle navigateur alternatif, il devient possible de le manipuler (taille, orientation…) et surtout d'afficher des éléments par-dessus chose impossible en utilisant le contrôle WebBrowser de base.

Cependant, cette méthode présente la limite précise qu'il s'agit d'une utilisation de composant non supporté et d'une utilisation de l'API AJAX de la plateforme Bing Maps for Enterprise dans un contexte non supporté.

Utilisation d'un navigateur alternatif et du contrôle Silverlight

Description technique :

En utilisant le contrôle navigateur basé sur Chromium à travers le wrapper dédié, on charge une application simple intégrant le contrôle Silverlight de la plateforme Bing Maps for Enterprise.

Le projet WPFChromium présenté avec la solution précédente permet de réaliser cette intégration mais peut amener des difficultés de chargement du contenu Silverlight.

Analyse :

Avantages	**Inconvénients**
Intégration graphique facilitée	Méthode non-supportée
Performances	Problème possible de compatibilité Silverlight
Possibilité d'intégrer la logique applicative dans l'application Silverlight	Evolution, difficulté de développement et limitations d'interopérabilité possibles

Mon avis :

Cette solution présente de nombreux points de doutes avec notamment un gros souci de compatibilité/support du contrôle navigateur tout comme le support Silverlight dans ce contexte.

La complexité de développement et les limitations possibles d'interopérabilité avec ce navigateur alternatif devront faire l'objet d'une reflexion approfondie.

Utilisation d'une application Silverlight directement en WPF

Description technique :

En réalisant une application Silverlight intégrant le contrôle Silverlight de Bing Maps for Enterprise, on utilise directement ce composant au sein de l'application WPF.

Pour cela plusieurs techniques possibles dont voici quelques liens utiles :

- http://msdn.microsoft.com/en-us/library/ff921109(v=pandp.40).aspx
- http://blogs.msdn.com/b/clrteam/archive/2009/12/01/sharing-silverlight-assemblies-with-net-apps.aspx
- http://karlshifflett.wordpress.com/2009/11/19/silverlight-3-4-library-sharing-with-net-4-0-library-or-wpf/
- http://msdn.microsoft.com/en-us/library/cc903925(VS.95).aspx
- http://jmorrill.hjtcentral.com/Home/tabid/428/EntryId/446/Differences-between-WPF-and-Silverlight-Rendering-Stacks.aspx

Un projet CodePlex facilite cette intégration, il s'agit de SilverlightViewport développé par Jeremiah Morrill (qui soit dit au passage est un vrai guru en WPF et autres technologies de présentation).
Ce projet est disponible via ce lien : http://silverlightviewport.codeplex.com/

Analyse :

Avantages	**Inconvénients**
Intégration WPF	Méthode non supportée
Performance du contrôle Silverlight	Complexité de développement
Possibilité d'intégrer la logique applicative dans l'application Silverlight	Dépendance au plugin Silverlight

Mon avis :

Cette technique d'implémentation est assez délicate à mettre en place et nécessite une connaissance pointue des technologies WPF et Silverlight.

Elle présente toutefois un avantage notable, elle permet en effet de s'intégrer de manière parfaite en WPF et n'impose pas de dépendance à un quelconque contrôle navigateur.

Utilisation du plugin 3d à travers le contrôle InfoStrat.VE – CodePlex

Description technique :

Cette solution technique repose sur l'utilisation du plugin 3d disponible avec la version 6.3 et qui est utilisé au sein d'une surface Direct3D directement au sein de WPF.

Le projet InfoStrat.VE disponible sur CodePlex et développé par InfoStrat, fournit la couche de manipulation entre WPF et le plugin 3D ce qui simplifie grandement la gestion des événements et la manipulation du composant natif.

Analyse :

Avantages	**Inconvénients**
Utilisation du plugin 3D	Fin de vie du plugin 3D
Performances	Méthode non supportée
Intégration WPF et simplicité du développement	Nombre d'instance simultanée limitée du contrôle

Mon avis :

Cette solution fonctionne à merveille et permets de proposer des performances élevées et une expérience 3d qui se combine à merveille sur les applications Surface.

Cependant, la fin de vie programmée du plugin 3D de la plateforme Bing Maps for Enterprise impose à chacun de considérer cette solution comme obsolète et même à remplacer dans un futur proche compte tenu de l'évolution prochaine de la plateforme, relativement incertaine vis à vis de ce plugin 3D.

Utilisation du contrôle GreatMaps for WPF – CodePlex

Description technique :

La solution technique employée utilise un contrôle natif WPF (ou WinForm et même Windows Mobile) qui repose sur un fonctionnement standard de génération de tuile.

Le contrôle est disponible au sein du projet GreatMap hébergé sur CodePlex rend possible l'affichage de cartes chargées sous formes de tuiles.

Voici le lien vers ce projet : http://greatmaps.codeplex.com/

Analyse :

Avantages	**Inconvénients**
Intégration WPF	Support limité
Simplicité du développement et possibilité de changement de fournisseur de contenu	Evolution délicate du contrôle vis à vis des services de la plateforme
Performances	

Mon avis :

Cette solution mérite le détour puisque dans la version actuelle, on dispose d'un contrôle dédié à WinForm et un autre à WPF.

Le contrôle permet d'utiliser de multiples fournisseurs de carte (Bing Maps for Enterprise, OSM, Google Maps…) et simplifie grandement le développement et l'intégration d'une carte performante dans vos applications WPF.

Utilisation d'un contrôle commercial dédié – Telerik RadMap

Description technique :

En utilisant le contrôle RadMap de la librairie proposée par Telerik, on obtient un composant cartographique multifournisseur, permettant d'afficher des cartes nativement en WPF.

telerik
deliver more than expected

Le contrôle tente de reproduire le comportement et le rendu disponible dans celui proposé en Silverlight reposant sur la technologie DeepZoom.

Analyse :

Avantages	**Inconvénients**
Méthode supportée par Telerik	Performances et expérience relativement limitées
Simplicité de développement et possibilité de changement de fournisseur de contenu	Composant commercial et licence associée
Intégration WPF	Evolution du composant

Mon avis :

Cette solution commerciale présente un contrôle simple à utiliser mais qui n'apporte pas une expérience utilisateur suffisante pour approcher celle du contrôle AJAX ou Silverlight.

L'aspect commercial de la solution n'est également pas là pour simplifier la mise en place au sein d'un projet client.

Développement personnalisé natif WPF

Description technique :

La solution la plus radicale consiste à recréer soi-même un contrôle en WPF natif utilisant des éléments standards pour charger une carte à travers un canvas dédié.

Analyse :

Avantages	**Inconvénients**
Maitrise complète du développement	Complexité de développement
Intégration WPF et performances	Coût lié
Evolution du contrôle vis à vis de la plateforme	Problématique d'évolution, de performances, de test et charges

Mon avis :

Certainement l'option la plus permissive peu importe l'exploitation du contrôle qui est faite, cette solution a cependant des inconvénients notables puisqu'elle nécessite un développement personnalisé qui peut très vite devenir couteux en temps et très délicat à appréhender.

Cas particuliers de la table Microsoft Surface

La table Surface de Microsoft se présente à présent en 2 versions distinctes que je choisis ici de présenter brièvement. Nous allons ensuite nous intéresser au cas spécifique des applications s'exécutant sur ce type de périphérique.

Le détail complet et des articles explicatifs précis sont disponibles sur la MSDN :
http://msdn.microsoft.com/en-us/library/ee804845(v=Surface.10).aspx

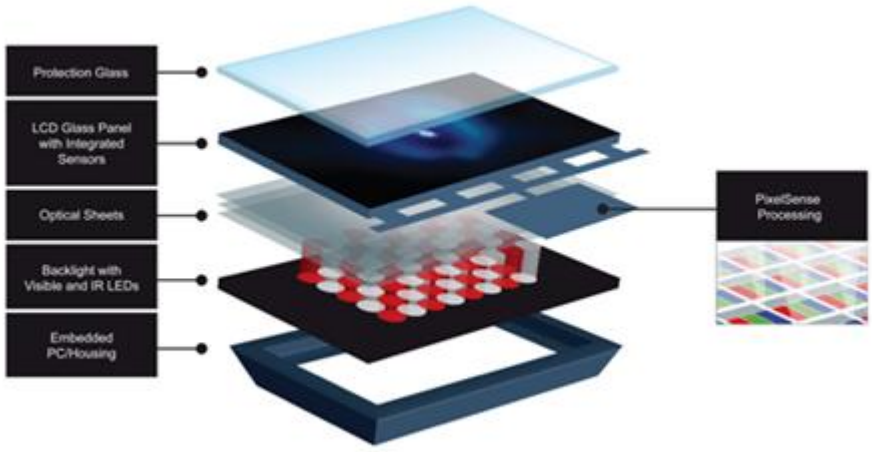
La table Surface en version 1 (Microsoft Surface Table V1)

La première version a été rendue disponible à partir de 2009 et reposait sur une technologie de reconnaissance de forme utilisant des caméras intégrées à la table qui se présente comme cela :

La table Surface en version 2 (Microsoft Surface Table V2)

La seconde version a été annoncée et devrait être commercialisée dans le second semestre 2011.

Pour réaliser la reconnaissance de forme et la détection des interactions utilisateurs, Samsung et Microsoft ont mis au point une technologie baptisée PixelSense qui s'appuie sur un réseau de LED Infrarouge et une couche de détection optique capturant le résultat réfléchie de l'infrarouge.



Pour plus d'informations sur la technologie PixelSense utilisée au sein de cette Surface 2.0, n'hésitez pas à vous rendre sur le site dédié : http://www.microsoft.com/surface/en/us/pixelsense.aspx

Principales différences

Forme, matériel et technologie :

La table Surface V1 pèse 70 Kg et utilise une technologie de reconnaissance de forme basé sur un réseau de caméra.

La table Surface V2 quant à elle présente un poids moindre (40 Kg) et donne la possibilité d'être orienté et même être disposée sur un mur. Une épaisseur de seulement une dizaine de centimètre grâce à l'utilisation de la technologie PixelSense.

Evidemment, la configuration matérielle (le PC qui exécute les applications) a aussi connu une cure de jouvence. Le système d'exploitation est logiquement changé pour passer de Windows Vista (v1) à Windows 7 (v2).

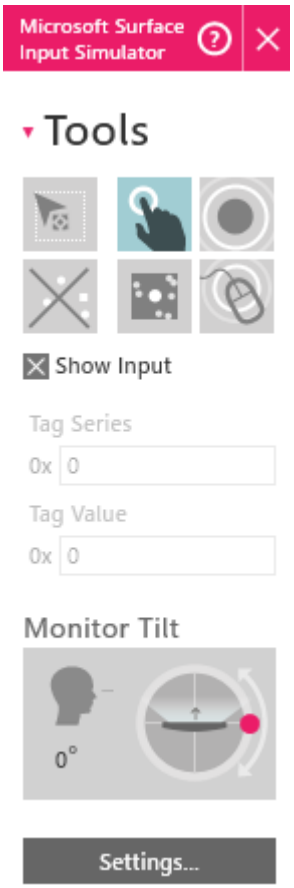
En résumé :

	Surface V1.0	**Surface V2.0**
Ecran	30" XGA projecteur avec surface polie	40" LCD avec une vitre de type Gorilla Glass
Résolution	1024 * 768	1920 * 1080 (HD 1080p)
Technologie multitouch	Réseau de projecteur et de 5 caméras 52 contacts simultanés	PixelSense 50 contacts simultanés
Dimensions	56 cm d'épaisseur 68.6 cm de large pour la surface 57.2 cm de large pour la base 108 cm de long	10 cm d'épaisseur 102 cm de diagonale
Poids	70 Kg	40 Kg
Disposition	Horizontale	Horizontale ou Verticale (inclinable)
Processeur	Intel Core 2 Duo (2.13 Ghz)	AMD Athlon II X2 Dual-Core Processor 2.9GHz
Carte graphique	ATI X1650	AMD Radeon HD 6700M
RAM	2 Go DDR2	4 Go DDR3
Disque dur	250 Go SATA 7200 t/min	320 Go SATA 7200 t/min
Support HDMI	Non	Oui
Système	Windows Vista Pro 32-bit	Windows 7 Pro 64-bit
Coût	12 500 $ / 9000 € env.	7 600 $ / 5 400 € env.

Sources :

- http://technet.microsoft.com/en-us/library/ee692114(Surface.10).aspx
- http://ctlabs.blogspot.com/2011/01/microsoft-surface-20-multi-touch-wonder.html
- http://en.wikipedia.org/wiki/Microsoft_Surface#Surface_2.0

Aspect logiciel :



Fondamentalement la technologie utilisée reste identique puisqu'il s'agit dans les 2 versions d'un Framework basé sur WPF. Il est également possible de réaliser des applications en XNA sur ce type de périphérique.

Le gros point de différence réside dans la manière de gérer les contacts qui dans la première version étaient des ContactEvent (avec des propriétés dédiées).

Le nouveau SDK ainsi que les outils de développement ou de simulation de la table Surface ont eux aussi été mis à jour avec notamment un simulateur amélioré de la table :

Enfin, évidemment, le Framework Surface a été étendu avec de nouveaux contrôles et nouvelles possibilités disponibles à travers ce lien :

http://www.microsoft.com/download/en/details.aspx?id=26716

Concernant la migration d'une application Surface v1 vers Surface v2, pour les cas simples, il est possible d'utiliser le PowerToy fourni par Microsoft et présenté à travers le lien suivant : http://msdn.microsoft.com/en-us/library/gg697146.aspx.

Considérations pour vos applications pour la table Surface V1.0

La table Surface v1.0 repose sur une couche de capture des événements qui s'opère au-dessus de l'application WPF. Il faut alors pouvoir capturer ses contacts au-dessus des composants utilisés et bindés ces derniers sur une manipulation WPF adaptée.

Les solutions reposant sur l'utilisation du contrôle **WebBrowser** ne peuvent pas fonctionner en manipulation directe de la carte : on ne peut donc pas capturer les contacts au-dessus d'un contrôle **WebBrowser** classique.
En effet, ce contrôle existe depuis les anciennes versions du Framework et repose sur un contrôle Win32 et le rendu s'opère au dessus de tout contrôle WPF et également au-dessus de la couche de capture des contacts. Dès lors il est donc impossible d'obtenir un événement de contact au-dessus de la page hébergé par le contrôle WebBrowser.

Evolution des développements

Les applications développées pour la table Surface seront les mêmes que ceux des tablettes Windows 7 ou des PC avec écran tactiles. Elles seront naturellement compatibles sur PC avec une simple souris.

Ce qui intéressant dans ce cas c'est la capacité simple permettant de migrer une application sur un type de support vers un autre et inversement. Il est également intéressant de voir que l'intégration de contrôles avancés se fera de manière plus aisée et surtout une méthode unique, les considérations concernant l'intégration de la cartographie en WPF se réduisent puisqu'il s'agit là des mêmes développements.

Attention toutefois, les applications s'exécutant sur une table Surface doivent, à mon sens, faire preuve d'une réflexion particulière sur l'utilisation qui va en être faite. Il faut concevoir l'application comme étant possiblement exploitable de n'importe quel côté de la table et surtout avec de l'information adaptée et lisible dans ce cas présent. L'utilisation simultanée par plusieurs utilisateurs est également un élément dont il faut tenir compte et qui est à réfléchir.

Conclusions

L'ensemble de ces solutions techniques d'intégration de plateforme cartographique au sein d'application WPF sont à considérer en fonction du contexte final d'exécution.

En effet, on ne peut pas aisément désigner LA solution parfaite pour tous les cas car là où pour certains le besoin en support et évolution du composant est primordial, l'intégration fine et le besoin de customisation seront négligés et inversement.

Par contre, la solution utilisant le plugin 3d (Concierge ou contrôle InfoStrat.VE) ne doivent plus être utilisées pour de nouveaux projets et devront très certainement subir une migration dans les mois à venir à la fin de vie du plugin (novembre 2011).

Il également à noter que dans ce type d'intégration, il est toutefois nécessaire de conserver en tête l'aspect légal et donc se rapprocher de Microsoft pour obtenir davantage d'information concernant les licences d'utilisation de la plateforme Bing Maps for Enterprise.

Autres pistes à explorer :

Si vous avez en tête d'autres solutions d'intégration de l'API Bing Maps ou plus généralement de cartographie au sein d'application WPF à laquelle je n'aurai pensé, n'hésitez pas à me contacter ou à laisser un commentaire.

Je n'ai pas présenté la solution d'intégration de Google Earth API puisque je voulais des solutions d'intégration de la plateforme Bing Maps for Enterprise principalement, mais il est évident que cela peut être une solution possible d'intégration.

La solution d'intégration ESRI n'a pas non plus été présentée puisque leurs contrôles ne sont pas librement accessibles en version d'évaluation.

Aussi, si vous aussi vous rencontrez ce genre de besoin, n'hésitez pas à me contacter pour échanger sur le sujet.