Cartographie et Localisation Géographique

Catégorie : VBA par l'exemple
Publié par myDearFriend! le 05-08-2007

Je souhaiterais disposer d'une carte géographique (ou routière) sur laquelle je pourrais visualiser des lieux précis, épingler des points repères, les gérer, les afficher et les commenter à ma guise... et le tout sur Excel !

C'est un sujet qui ne manque pas d'intérêt, d'autant qu'une vraie solution permettrait de construire des applications originales au fin de présentation ou de gestion de tout ordre (gestion de clientèle, gestion de parc automobiles ou de matériel, etc...). Attention toutefois au respect des droits d'utilisation en cas d'applications commerciales notamment...

Quels sont les éléments de base dont j'ai besoin ?

1. Une base de données représentant les points à afficher (une liste d'adresses par exemple)
2. Une carte suffisamment précise et ciblant une étendue géographique particulière.

Si j'obtenais ces prérequis, quels sont les outils ou fonctionnalités dont j'aimerais disposer ?

1. Pouvoir faire un zoom sur la carte pour en découvrir les détails, ou à l'inverse, accroître son champ de vision.
2. Mieux encore, parcourir la carte en la faisant glisser sans aucune limite.
3. Atteindre un lieu précis en saisissant simplement une adresse postale.
4. Atteindre ce même lieu également en saisissant les coordonnées GPS si je les connais.
5. Positionner des repères personnalisés là où je le souhaite et les afficher ou les masquer à volonté.

Qui d'entre nous n'a jamais fait une recherche d'itinéraire sur des sites web comme ViaMichelin ou Mappy par exemple ?
Vous l'avez deviné... c'est ce type de carte qui conviendrait à ce projet !

Mais peut-on obtenir l'équivalent sur Excel et le piloter par VBA ?

La réponse est oui ! Mais il convient toutefois de noter que :

1. une connexion internet reste nécessaire.
2. l'utilisation d'un contrôle WebBrowser est requise, ce qui rend donc incompatible l'application avec l'environnement Mac.
3. pour le développement, quelques notions des langages HTML et JavaScript sont les bienvenues...

Par ailleurs, je précise que le présent article s'adresse à celles et ceux qui maîtrisent déjà un peu VBA et qui connaissent l'utilisation des Userforms en particulier.

Virtual Earth ou la Cartographie selon Microsoft

Présentation de l'outil et préparatifs

A l'instar de Google avec Google Maps, Microsoft met à disposition de tous, Virtual Earth, un outil de localisation géographique développé pour ses services "Windows Live". C'est cet API contrôle de carte qui a attiré mon attention. Les cartes présentées souffrent toutefois d'un léger défaut : elles sont en version anglophone. Bien que ce ne soit pas trop gênant, lorsqu'on observe l'hexagone par exemple, on peut y voir ci et là, trois ou quatre noms de nos régions françaises traduits dans la langue de shakespeare...

Voici une démonstration fonctionnelle de Virtual Earth

vous pouvez naviguer sur la carte ci-dessous à l'aide la souris et zoomer avec la roulette ou en cliquant sur +/-

(il semble que les utilisateurs de Mac ne puissent voir cette démonstration que sous FireFox)

Par programmation, il est ensuite possible d'afficher des points repères personnalisés avec bulle d'info,
comme dans l'illustration ci-dessous.

On peut obtenir également la même chose en vue aérienne.

Virtual Earth en 3D ?
Dans le navigateur intégré par défaut (il n'apparait pas ci-dessus !), si vous cliquez sur le bouton 3D, vous serez invité à télécharger la version beta du contrôle ActiveX Virtual Earth 3D. A vous de voir si vous souhaitez ou non tester cette version beta. Je l'ai testé moi même, mais en toute franchise, j'avoue qu'à ce jour Google Earth garde toutefois une bonne longueur d'avance sur le sujet...

Cet outil de localisation géographique possède de très nombreuses possibilités en terme de paramétrages, de fonctions et méthodes disponibles. Pour cet article, je me contenterai d'en présenter uniquement les fonctions de base, je vous laisse explorer et tester le reste (je n'ai d'ailleurs pas encore tout exploré moi-même)...

Si vous possédez quelques notions HTML et Javascript, je vous conseille vivement de consulter le Virtual Earth Interactive SDK ainsi que Virtual Earth Map Control 5.0 de la library MSDN.

De part la conception de ce contrôle, l'affichage ainsi que le paramétrage de Virtual Earth sont gérés par JavaScripts. Seules quelques lignes de code HTML et JavaScript suffisent à assurer la connexion avec les scripts de contrôle Virtual Earth en ligne et donc à permettre ensuite le pilotage de l'outil par programmation.

Bien que mon objectif soit d'afficher les cartes géographiques directement dans un WebBrowser et de créer un projet VBA, le fichier d'initialisation HTML reste toutefois obligatoire et il faudra donc le créer sur le disque dur.

Je réduis cependant ce fichier HTML au strict nécessaire, et toutes les commandes dont j'aurai besoin pour mener à bien le projet pourront être générées par la suite, "à la volée", directement par code VBA.

1. J'ouvre donc le Bloc-Note de Windows et je copie les lignes d'instructions HTML et JavaScript suivantes :

   Le fichier d'initialisation HTML qui permet d'afficher les cartes dans le WebBrowser

```
      src="http://dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=5">

         var map = null;

         function GetMap()
         {
            map = new VEMap('myMap');
            map.LoadMap();
         }


   onload='GetMap();'>


id='myMap' style='position:relative; width:640px; height:540px;'>
```

1. Je sauvegarde le fichier sous le nom "ve_connexion" dans le répertoire créé spécialement pour mon projet.

1. Je ferme le Bloc-Note et je change l'extension .txt du fichier en .html.

ASTUCE

Au lieu de créer manuellement ce fichier "ve_connexion.html" sur le disque dur, il est également possible de le générer directement à l'aide de VBA à l'ouverture du classeur Excel (création dans le répertoire de l'application ou dans un répertoire temporaire), puis de le supprimer en fin de session. Cette solution a l'avantage de permettre la distribution d'un seul et unique fichier XLS, mais aussi d'être certain que le fichier d'initialisation HTML sera bien présent au bon endroit. Le travail devient ainsi plus propre et plus sûr.

J'utilise d'ailleurs cette solution dans l'application mDF XLmap développée en parallèle de cet article...

Les préparatifs sont maintenant terminés, je vais pouvoir me consacrer au classeur Excel et son projet VBA.

Le Classeur Excel et son projet VBA

Dans le répertoire spécialement réservé pour ce projet, je vais maintenant créer un classeur Excel nommé "ExempleVirtualEarth.xls".

Direction l'éditeur VBE (ALT + F11), je vais construire l'interface de mon application.

Dans le projet, j'insère donc un Userform, les contrôles et propriétés suivants :

Contrôles	Name	Caption		Left	Top	Width	Height
Userform	Userform1	Exemple d'utilisation de l'API Virtual Earth	-	-	650	445	
WebBrowser	WebBrowser1	-	6	6	500	410	
CommandButton	btnZPlus	Zoom +	516	162	54	18	
CommandButton	btnZMoins	Zoom -	588	162	48	18	
CommandButton	btnCentrer	Centrer la carte (sur France)	516	186	120	18	
CommandButton	btnAjout	Ajouter un repère au centre	516	258	120	18	
CommandButton	btnRaz	Effacer les points repères	516	282	120	18	
CommandButton	btnAdr	Atteindre une adresse	516	318	120	18	
CommandButton	btnQuit	Quitter	516	396	120	18	
CheckBox	chkNav	Navigateur intégré	522	216	90	15	

CheckBox	chkMini	Mini carte	522	234	90	15
OptionButton	optKms	Echelle en Kilomètres	522	348	105	15
OptionButton	optMil	Echelle en Miles	522	366	105	15

Ce qui devrait donner quelque chose comme ceci :

Le code VBA

En premier, je crée une procédure pour charger et afficher le Userform et je l'associe à un bouton (ou une image) placé sur la feuille de calcul. J'utilise une simple fonction Dir() pour vérifier la présence du fichier d'initialisation HTML (qui devrait être dans le même dossier que le classeur Excel). Cette procédure contient le code suivant :

```
Dim FichierHtml As String

    FichierHtml = ThisWorkbook.Path & "ve_connexion.html"

    'Vérifier l'existence du fichier d'intialisation HTML avant lancement
    If Dir(FichierHtml) <> "" Then
        UserForm1.Show
    Else
        MsgBox "Le fichier """ & FichierHtml & """ est introuvable !" _
                & vbLf & vbLf & "Opération annulée.", vbOKOnly, "myDearFriend!"
    End If
```

Le module de code du Userform :

Dans le module de code du Userform cette fois, je crée l'évènement Initialize(). C'est ici que je fais le lien entre le contrôle WebBrowser et l'API Virtual Earth. Au chargement du USF, la méthode Navigate va lire le fichier d'initialisation Html et lancer ainsi la connexion avec le contrôle de carte :

```
Private Sub UserForm_Initialize()
    'Connexion au contrôle de carte en ligne
    WebBrowser1.Navigate ThisWorkbook.Path & "ve_connexion.html"
End Sub
```

Je le disais plus haut : « l'API Virtual Earth se pilote à l'aide de commandes ou fonctions JavaScript ».

Pour lancer des fonctions JavaScript depuis VBA, il convient d'utiliser la méthode DHTML execScript qui s'applique à l'objet window (dans le cas présent, cet objet sera la fenêtre contenant le document web).

Cela se fait de cette façon :

La méthode execScript

```
WebBrowser1.Document.parentWindow.execScript "LaFonction();", "Javascript"
```

LaFonction(); est ici une chaîne de caractères représentant la fonction JavaScript à exécuter.

Comme je vais devoir lancer plusieurs fonctions JavaScript dans cette application, et pour clarifier le code VBA au maximum, je crée une procédure qui assurera la bonne syntaxe lors de chaque appel :

```
Private Sub EnvoiScript(Js As String)
    WebBrowser1.Document.parentWindow.execScript Js, "Javascript"
End Sub
```

Ainsi, lorsque je voudrai lancer une Fonction() JavaScript depuis mon module de code, il me suffira simplement d'appeler la procédure EnvoiScript et de lui passer la chaîne de commande en argument, comme ceci :

```
EnvoiScript "LaFonction();"
```

Je m'occupe maintenant des évènements associés à chaque contrôle du Userform.

Je commence par les contrôles CommandButton :

- Tout d'abord les boutons "Zoom+" et "Zoom-"
  Il convient simplement de faire appel aux méthodes ZoomIn() et ZoomOut() du contrôle de carte Virtual Earth.

```
Private Sub btnZPlus_Click()
    EnvoiScript "map.ZoomIn();"
End Sub

Private Sub btnZMoins_Click()
    EnvoiScript "map.ZoomOut();"
End Sub
```

- Le bouton "Centrer la carte"
  Nécessite l'appel à la méthode SetCenterAndZoom() pour afficher une zone particulière de la carte dans le WebBrowser. Ici, je m'arrange pour que le territoire représentant la France métropolitaine soit affiché au mieux et de façon centrée. J'indique les coordonnées Latitude/Longitude ainsi que l'indice de zoom souhaités :

```
Private Sub btnCentrer_Click()
    'Permet de re-centrer l'affichage de la carte dans le WebBrowser sur

'des coordonnées Latitude/Longitude prédéterminées et de définir un zoom (vale
ur de 1 à 19)
    EnvoiScript
"map.SetCenterAndZoom(new VELatLong(47.010225655683485, 2.3291015625000036), 6
);"
End Sub
```

ASTUCE

Pour trouver rapidement les coordonnées Latitude/Longitude souhaitées sans avoir à programmer quoi que ce soit à cette fin :

Un petit tour du côté du Virtual Earth Interactive SDK. J'affiche la carte de France approximativement au centre de la zone de carte, puis je développe l'option Get map info dans le menu à gauche et je clique sur Get center LatLong. Il ne me reste plus qu'à noter les coordonnées qui s'affichent à l'écran !

Par défaut, lors de la connexion le contrôle de carte Virtual Earth présente le territoire américain. Peut-être voudriez-vous voir afficher l'hexagone dès l'ouverture du Userform au lieu des Etats-Unis ? Dans ce cas, plusieurs solutions peuvent être envisager, mais la solution la plus simple reste d'insérer directement les coordonnées Latitude/Longitude souhaitées dans la méthode LoadMap() que l'on trouve dans le fichier d'initialisation Html.

Dans ce fichier, on pourra donc remplacer :
map.LoadMap() par map.LoadMap(new VELatLong(47.010225655683485, 2.3291015625000036), 6);

- Le bouton "Ajouter un repère au centre"
  L'objectif ici est d'épingler au centre de la carte un point repère et sa bulle d'info. Pour cette tâche, il va falloir utiliser plusieurs lignes d'instructions cette fois.
  L'insertion se fait grâce à la méthode AddShape(shape) . Il convient toutefois de créer en amont une instance de l'objet Shape à insérer et de définir le Titre ainsi que la Description (Texte) de la bulle d'info correspondante. Le code VBA sera celui-ci :

```
Private Sub btnAjout_Click()
Dim T As String
    'Ajouter un point repère (Pushpin) au centre de la carte
    T = "var shape = new VEShape(VEShapeType.Pushpin, map.GetCenter());" _
        & "shape.SetTitle('Exemple de point repère');" _
        &
"shape.SetDescription('Ceci est le symbole de point repère par défaut "_
        & "fourni par Virtual Earth');" _
        & "map.AddShape(shape);"
    EnvoiScript T
End Sub
```

- Le bouton "Effacer les points repères"
  Cette opération est réalisée par la simple méthode DeleteAllShapes() :

```
Private Sub btnRAZ_Click()
    'Supprime tous les repères "épinglés" sur la carte
    EnvoiScript "map.DeleteAllShapes();"
End Sub
```

- Le bouton "Atteindre une adresse"
  Voici une option qui rend l'API Virtual Earth vraiment intéressant. Un clic sur ce bouton et l'utilisateur est invité à saisir une adresse postale (avec toutefois une certaine syntaxe à respecter). La méthode Find() du contrôle de carte va ensuite interroger la base de données de Virtual Earth et afficher automatiquement le lieu géographique visé ou un message d'alerte en cas d'échec (si l'adresse manque de précision par exemple) :

```
Private Sub btnAdr_Click()
Dim T As String

'Rechercher et afficher un lieu en fonction de l'adresse saisie dans une Input
Box
    T =
"Saisissez une adresse complète pour optimiser les chances de résultat. Les él
éments " _
        & vbLf & "[Numéro et nom de rue] , [Code Postal et Ville] , [Pays]" _
        & vbLf &
"doivent être saisis dans l'ordre et séparés par une virgule." _
        & vbLf & vbLf & "Exemple :" _
        & vbLf & "7 rue de la liberté,21000 dijon,france"
    T = InputBox(T, "Atteindre un lieu...")

    If T <> "" Then

'Attention, ici l'apostrophe remplace les guillemets dans les instructions Jav
aScript.

'Aussi, si l'adresse saisie contient une apostrophe, il convient de faire préc
éder cette
        '
dernière par un caractère  pour éviter un plantage du script. Et pour assurer
une

'compatibilité avec Excel97, j'utilise la fonction de feuille de calcul Substi
tute()
        'au lieu de la fonction Replace().
        T = Application.Substitute(T, "'", "'")
        'On emploie la méthode "Find()" pour rechercher l'adresse.
        'Syntaxe :

'VEMap.Find(what, where, findType, shapeLayer, startIndex, numberOfResults,

'showResults, createResults, useDefaultDisambiguation, setBestMapView, callbac
```

```
k);
        EnvoiScript "map.Find(null, '" & T & "');"
    End If
End Sub
```

- Le bouton "Quitter"
  Je ne pense pas qu'un commentaire soit nécessaire ici...

```
Private Sub btnQuit_Click()
    Unload Me
End Sub
```

Je passe ensuite aux contrôles de type Checkbox :

- La case à cocher "Navigateur intégré"
  L'affichage du navigateur intégré à Virtual Earth peut être désactivé à la demande. Pour ce faire, les méthodes ShowDashboard() et HideDashboard() sont nécessaires :

```
Private Sub chkNav_Click()
    'Affiche ou supprime le navigateur Web interne
    EnvoiScript IIf(chkNav.Value, "map.ShowDashboard();",
"map.HideDashboard();")
End Sub
```

- La case à cocher "Mini carte"
  Le contrôle de carte Virtual Earth permet d'ajouter une mini-carte dans la fenêtre pour aider à la navigation. Son affichage sera géré par les méthodes ShowMiniMap() et HideMiniMap() :

```
Private Sub chkMini_Click()
    'Affiche ou supprime la mini carte de navigation
    EnvoiScript IIf(chkMini.Value, "map.ShowMiniMap();", "map.HideMiniMap();")
End Sub
```

Je termine maintenant par les deux derniers contrôles de type OptionButton du Userform.

- Les boutons d'option "Echelle en Kilomètres" et "Echelle en Miles"

L'unité de distance utilisée dans l'échelle (visible au bas et à droite de la carte dans le WebBrowser) peut être définie en kilomètres ou en miles. La définition de cette unité de distance sert également à d'autres options et méthodes disponibles avec cet API Virtual Earth, mais je m'arrêterai là pour cet article.

```
Private Sub optKms_Click()
    EnvoiScript "map.SetScaleBarDistanceUnit(VEDistanceUnit.Kilometers);"
End Sub

Private Sub optMil_Click()
    EnvoiScript "map.SetScaleBarDistanceUnit(VEDistanceUnit.Miles);"
End Sub
```

Eh bien, mon projet de test est maintenant terminé !

Evidemment, ces quelques procédures et méthodes ne représentent qu'une petite partie de ce qu'il est possible d'obtenir avec un tel outil. Mais personnellement, je trouve déjà ça assez bluffant !

Persuadé de l'intérêt de ce projet, j'espère vous avoir donné envie de poursuivre l'investigation sur ce sujet. Consultez et testez la documentation mise à votre disposition (voir les liens plus haut dans cet article).

Pour ma part, c'est décidé, je vais continuer à travailler dans cette voie qui, sur un support Excel, ouvre des perspectives innovantes et vraiment très intéressantes...

Cet article décrit en détail la création d'un projet Excel-VBA de test de géolocalisation grâce au contrôle de carte Virtual Earth de Microsoft.

Vous trouverez en Section Téléchargements et plus particulièrement dans la nouvelle catégorie Cartographie et Localisation Géographique le fichier construit précédemment :

mDF Exemple Virtual Earth : le projet décrit dans cet article et le code VBA correspondant accessible.

Vous pouvez également télécharger en catégorie Utilitaires, une application complète réalisée en parallèle de cet article (nul doute que d'autres versions suivront!). Elle met en pratique l'enseignement tiré de mes recherches sur le sujet :
mDF XLmap v2.0 : cette application permet de gérer une base d'adresses, de localiser géographiquement vos contacts et plus encore...