Les Applications Windows

I. Introduction :

1. Mode Console :

l'interface Homme/Machine est constituée exclusivement par :

- Le clavier vue comme un fichier séquentiel.
- L'écran vu comme un fichier de sortie séquentiel.

Avantages : programmation de l'interfaces H/M très simple, adaptation très simple d'un OS à un autre.

Inconvénients : peu ergonomique, obsolète : à réserver aux applications n'ayant pas E/S, un seul programme à accès à l'entrée clavier.

2. Mode Windows :

l'interface H/M est constituée par un paradigme d'utilisation introduit par Apple avec le Macintosh de :

- un affichage graphique.
- une interface de pointage : la souris.
- un clavier.

Avantages : assez ergonomique, simplicité d'utilisation, partage des ressources d'E/S IHM.

Inconvénients : complexité de la programmation (nécessite l'utilisation d'une API), adaptation à un autre OS très difficile car l'API est propre à l'OS cible.

API : interface de programmation d'application, ensemble de librairie contenant toutes fonctions, classes, Objets, définitions permettant de programmer des applications pour un domaine particulier. Ici pour la programmation Windows.

II. Fonctionnement de Windows :

1. Evénements :

Tous les programmes Windows se partagent les trois ressources du paradigme IHM.
C'est l'OS qui met tous ce petit monde d'accord.
Cela veut dire qu'un programme n'a jamais la main !

Le dialogue entre OS et les programmes s'opère aux travers de messages appelés **événements**.

NB : d'ailleurs quand un programme se plante l'OS affiche le texte « ne répond pas » car le programme ne répond plus aux événements générés par l'OS.
Notre programme est donc constitué au minimum d'une boucle principale charger de recevoir et d'envoyer des événements.

Début
Tant que le programme tourne faire
    début
    Attendre un événements
    Si (l'événement est prise en charge) alors Réagir à l'événement
    Fin
Fin

Chaque événement comporte :

- Le numéro du destinataire (handle).
- le type de l'événement : WM_ nombre entier.
- les données du message :
  - Paramètre 1 : entier 16 bits.
  - Paramètre 2 : entier 32 bits.
- L'heure à laquelle se produit l'événement.
- Coordonnées de la souris.

Avec Windows, la boucle principale est contenu dans fonction nommée « WinMain » apppelée au lancement du programme.

En C# avec .NET framework, la boucle principale est appelé par la méthode « Main » de l'objet « Program ». Il s'agit de la méthode « Application.Run » .

Exemple :

- Message envoyé par la souris :

  ```
  WM_MOUSEMOVE
  fwKeys = wParam;        // key flags
  xPos = LOWORD(lParam);  // horizontal position of cursor
  yPos = HIWORD(lParam);  // vertical position of cursor
  ```

- Message envoyer par le clavier lorsque l'on appuye sur une touche :

  ```
  WM_KEYDOWN
  nVirtKey = (int) wParam;    // virtual-key code
  lKeyData = lParam;          // key data
  ```

- Message de notification d'un menu ou d'un contrôle : WM_COMMAND

```
wNotifyCode = HIWORD(wParam); // notification code
wID = LOWORD(wParam);         // item, control, or accelerator identifier
hwndCtl = (HWND) lParam;      // handle of control
```

- message de fin d'application :

```
WM_QUIT
nExitCode = (int) wParam;   // exit code
```

Démo de « WinSight »

2. Focus clavier : lorsque une fenêtre est désignée pour recevoir les événements clavier.
3. Fenêtre Active : c'est la fenêtre que l'utilisateur actionne.
   La couleur de la barre de titre indique si elle est active.
   Cette fenêtre est toujours au premier plan.
   Pe on peut activer les menus de cette fenêtre.

4. Z order : les fenêtres sont supperposés suivant cette ordre.

III. Divers Objets Windows

On illustrera chaque classe d'objet par un exemple.

1. Fenêtre = Form (ulaire) :

Créer un nouveau formulaire avec Visual Studio C#. Ouvrir Program.cs
Repérer la fonction Main et l'appel de la boucle principale Application.Run()
(faire atteindre la définition et lire les commentaires).
Eléments d'une fenêtre :

a) Barre de titre : Menu système et Icônes Systèmes.
Déclenchent des événements systèmes.
b) Etat de la fenêtre au démarrage ou à la création.
c) Style de la fenêtre.
d) Menu :
Placer un composant MenuStrip.

Double cliquer sur Close : ainsi vous ajoutez une nouvelle méthode répondant à l'événement correspondant au menu (càd WM_COMMAND avec l'ID du menu mais ceci est une autre histoire).
Fermer le formulaire à l'appel de ce menu.

e) Fenêtres filles :

2. Boîte de dialogue : c'est une fenêtre fille particulière.
   a) Modale : A l'appel du menu Fermer, on voudrait demander une confirmation. Il suffit d'ajouter le code :

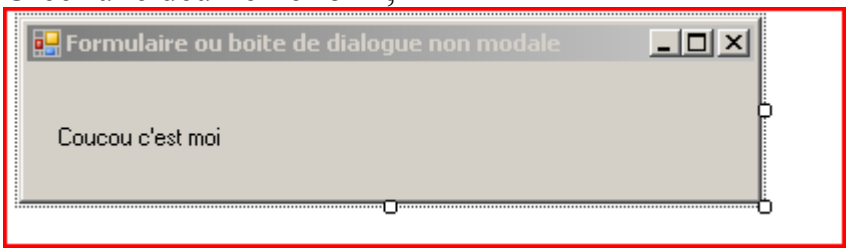
```
if (MessageBox.Show(" Etes vous sûre ?", "Confirmation",   MessageBoxButtons.YesNo )
                                                          ==DialogResult.Yes)
        Close();
```

Ce code créer et montre une boite de dialogue Modale : elle bloque l'exécution de l'application tant que l'on ne l'a pas refermée.
En C#, une boîte de dialogue est un formulaire qui renvoie un résultat appelé avec la méthode **ShowDialog**.
Il faut donc :

- Créer une instance de cette forme dans la fenêtre principale
- L'afficher avec ShowDialog.
- Traiter éventuellement la valeur renvoyée.

b) Non modale : ne bloque pas l'application (par exemple si l'on veut afficher des informations en permanence).

Créer une deuxième form,



Créer une instance de cette forme dans la fenêtre principale
L'afficher avec Show.

Modifier les propriétés : FormBorderStyle, ControlBox, MinmizeBox, MaximiseBox

c) Exemple de boite de dialogue :
- AboutBox :
- OpenFileDialog
- SaveFileDialog
- PrintDialog
- ...

3. Les Contrôles : ce sont des petites fenêtres filles d'un formulaire ou d'une boîte de dialogue. Ces objets ont en général un comportement simple.

a) Boutons : quand on appuye dessus déclenche un événement BN_CLICKED qui est en général géré par une méthode de la fenêtre parente « button_Click ».
Rajouter un bouton pour fermer la boîte de dialogue non modale de 2)b).

b) Texte :
c) Editeur de texte :
d) Liste :
e) Liste déroulante :
f) ...
g) Contrôle personnel : on peut créer nos propres contrôles

4. Les événements : ils correspondent aux messages Windows.

Exemple : événements souris :
Message Windows : WM_MOUSEMOVE
OnMouseMove : méthode déterminant le comportement de la fenêtre quand un événement MouseMove se produit.

Ajouter une méthode gérant l'événement :
Afficher les coordonnées de la souris dans la boîte non modale en ajoutant ce code :

```
protected override void OnMouseMove(MouseEventArgs e)
    {
      if (form2!=null && !form2.IsDisposed)   // le formulaire doit existé et ne pas avoir disparu
          form2.SetText( Convert.ToString( e.X)+ " " + Convert.ToString( e.Y));
     //  base.OnMouseMove(e);  // traitement habituel ??
    }
```

5. Les ressources : certaines constantes tel que les textes affichés ou l'apparence des fenêtres doivent parfois pouvoir être modifier sans avoir à recompiler l'application.

Ces constantes seront placées dans un fichier de ressources.Les ressources sont liées aux programmes au tout dernier moment par un « lieur de ressources ». Certains utilitaires, permettent d'ailleurs de modifier les ressources de l'exécutable d'une application.

(pe : placer dans les ressources tout les textes pour faciliter la traduction d'une application dans une autre langue.)

Avec C#, les ressources sont contenus dans les fichiers d'extension .resx et sont au format XML.

Ajouter une ressource Texte et l'utiliser pour le texte dans le formulaire 2b)

```
public Form2()
{
  InitializeComponent();
  label1.Text = Properties.Resources.String1;
}
```

a) Icône : changer l'icône du formulaire.

6. La sérialization :