Université de la Rochelle INFO-12605C - IHM
TP9 : Diaporama avec Windows Forms

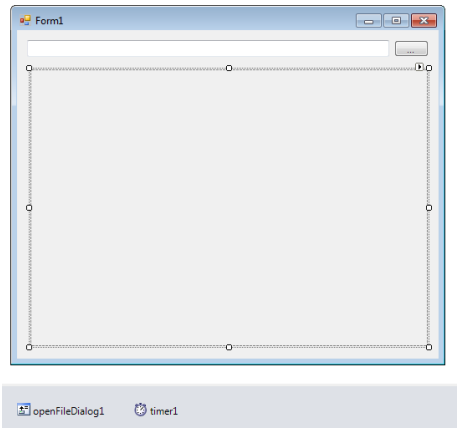
© B. Besserer, R. Péteri

Année universitaire 2015-2016

Lors de ce TP, vous allez réaliser une application qui affiche un diaporama à partir des images qui résident dans un répertoire. Outre l'ouverture et l'affichage d'images, on utilisera un objet Timer, on fera usage de la transparence, et l'interaction s'effectuera grâce à des touches clavier : l'affichage du diaporama en mode plein écran interdit en effet l'usage des contrôles IHM.

1 Mise en place du projet

Créez un projet Windows Forms. Dans la Form, placer une zone de texte, un bouton, un contrôle de type PictureBox, une boite de dialogue de type OpenFileDialog et un Timer (cet objet est disponible dans la boite à outils, section "composants"). Récupérez depuis moodle le fichier images.zip, et le fichier applescreensaver.zip. Copiez les images `1.jpg` à `7.jpg` qui se trouve dans l'archive images.zip dans un répertoire local, par exemple `D:/images`.



2 Affichage d'images

2.1 Affichage simple et affichage séquentiel

Il est très facile d'afficher des images car il existe un contrôle dédié : pictureBox. De plus, l'environnement .NET intègre les méthodes capable de lire ou d'écrire des formats d'images comme le JPG ou le PNG.
Lors du clic sur le bouton [...], la boite de dialogue doit permettre de parcourir l'arborescence et ouvrir le fichier sélectionné. On mettra en place un filtre (propriété de la boite de dialogue) afin de ne pouvoir sélectionner que les fichiers de type image (copier la chaine suivante dans le propriété Filter :
`Image Files (BITMAP,JPEG,PNG)|*.bmp;*.jpg;*.png`).
Si l'on sélectionne un fichier et que l'on valide, ce fichier sera ensuite ouvert et affiché dans la PictureBox (`pictureBox1.Image = Bitmap.FromFile(filename);`). le nom du fichier `filename` est évidemment récupéré d'une propriété de la boite de dialogue. et le nom du fichier sera affiché dans la TextBox.
Si seule une partie de l'image est visible, rapellez-vous des valeurs possibles de la propriété SizeMode du contrôle PictureBox (vu en TD)

Elements de correction : *La propriété sizemode du contrôle doit être placée sur "Zoom" (mise à l'échelle sans déformation) ou "Strech". Dans le code ci-dessous, il n'y a aucun test pour voir si le fichier est un fichier image.*

```
private void button1_Click(object sender, EventArgs e)
{
  String filename = String.Empty;
  if ((openFileDialog1.ShowDialog()) == DialogResult.OK)
  {
    textBox1.Text = filename = openFileDialog1.FileName;
    pictureBox1.Image = Bitmap.FromFile(filename);
  }
```

```
}
```

Testez ce fonctionnement, et lorsque celui ci est fonctionnel, modifier le parcours de l'arborescence en autorisant la sélection d'un répertoire : pour cela, retirez la boite de dialogue OpenFileDialog et remplacez par folderBrowserDialog. Récupérez le nom du répertoire.
Utilisez les classes DirectoryInfo et FileInfo pour parcourir ce répertoire (voir TD7, 1.1). Faites un appel à DirectoryInfo.GetFiles() et parcourez la liste des fichiers avec une structure de contrôle foreach. Affichez chaque image (**assurez-vous de ne récupérer que des fichiers de type image : jpg, bmp, png**) dans le contrôle pictureBox, rafraichissez l'affichage avec `pictureBox1.Refresh();` (`Refresh()` = `Invalidate()` suivi de `Update()`)et mettez un temps d'attente de 0.5 seconde entre chaque affichage d'image (`System.Threading.Thread.Sleep(int duration)`)

Elements de correction : *Dans le code ci-dessous, il n'y a aucun test pour voir si le fichier est un fichier image.*

```
/* ETAPE 2 : affichage sequentiel de toutes les images dans un repertoire */
private void button1_Click(object sender, EventArgs e)
{
  String filename = String.Empty;
  if ((folderBrowserDialog1.ShowDialog()) == DialogResult.OK)
  {
    System.IO.DirectoryInfo di = new System.IO.DirectoryInfo(folderBrowserDialog1.SelectedPath);
    foreach (System.IO.FileInfo fi in di.GetFiles())  // on récupère tous les fichier
    {
      if ((fi.Extension == ".bmp") || (fi.Extension == ".png") || (fi.Extension == ".jpg"))
      {
      pictureBox1.Image = Bitmap.FromFile(fi.FullName);
      pictureBox1.Update();
      System.Threading.Thread.Sleep(500);
      }
    }
  }
}
```

2.2 Utilisation d'une liste et d'un timer

Sur la Form, enlevez la TextBox, et remplacez-la par une listBox. Déclarez une donnée membre de type liste de String `List<String>`, qui contiendra les noms complets des images (chemin + nom de fichier) trouvées dans le répertoire. Dans une première structure de contrôle foreach (celle utilisée dans le code précédent), remplissez la liste (donnée membre) et en même temps ajoutez la chaîne au contrôle de type liste (`ListBox.Items.Add()`). La ListBox et l'instance de `List<String>` semble faire double usage. Mais à terme, l'objet de type ListBox pourra être supprimé de l'interface.

Ensuite, nous allons utiliser le timer pour cadencer le passage d'une image à la suivante. Le timer peut être créé par programme ou en glissant l'objet Timer depuis la boite à outils sur votre Form. Réglez le timer sur une période de 1 seconde (dans le panneau de propriétés en cliquant sur le symbole de l'objet timer, entrez 1000 (millisecondes) pour l'intervalle).

Associez une méthode à la méthode déléguée Timer.Tick. Cela peut se faire en sélectionnant l'objet Timer dans la vue [design], puis visualiser les événements de l'objet timer et d'écrire une méthode en réaction à l'événement *Tick*.

Le fonctionnement souhaité est le suivant :
- Un bouton supplémentaire "start" permet d'arrêter/de démarrer le timer. Une variable membre booléene doit refléter l'état de fonctionnement. Lorque le timer est en route, le bouton doit montrer "Stop", puis de nouveau "Start" si le timer a été arrêté.
- Lorsque l'événement tick est généré, la méthode déléguée va simplement incrémenter un itérateur (Pensez à utiliser des variables membres), charger l'image suivante de la liste et l'afficher. **Mais il faut boucler : si la dernière image de la liste est atteinte, on recommence**.
- la sélection d'une image dans le contrôle ListBox doit afficher immédiatement celle-ci et arrêter le timer. Lorsqu'on relance le timer avec le bouton, la séquence reprend à partir de l'image affichée.

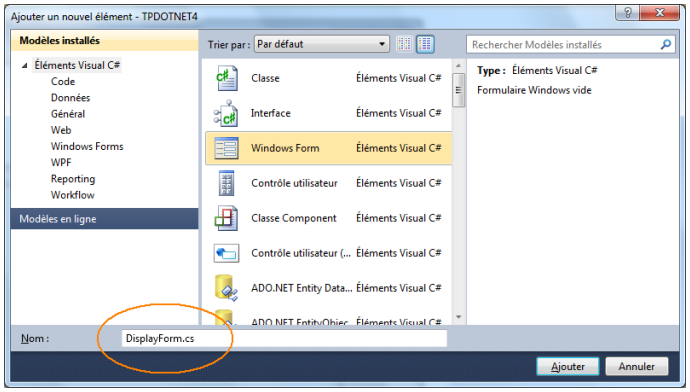
Montrez le fonctionnement et le code source (principalement la méthode déléguée `Timer1_Tick()`)

Elements de correction :

```
private void button1_Click(object sender, EventArgs e)
{
    String filename = String.Empty;
    if ((folderBrowserDialog1.ShowDialog()) == DialogResult.OK)
    {
System.IO.DirectoryInfo di = new System.IO.DirectoryInfo(folderBrowserDialog1.SelectedPath);
foreach (System.IO.FileInfo fi in di.GetFiles())  // on récupère tous les fichier
{
    if ((fi.Extension == ".bmp") || (fi.Extension == ".png") || (fi.Extension == ".jpg"))
    {
m_filenameList.Add(fi.FullName);
listBox1.Items.Add(fi.FullName);
    }
}
    }
}

private void timer1_Tick(object sender, EventArgs e)
{
    if (m_filenameList.Count > 0)
    {
pictureBox1.Image = Bitmap.FromFile(m_filenameList[m_iterator]);
pictureBox1.Update();
m_iterator++;
if (m_iterator >= m_filenameList.Count)
    m_iterator = 0;
    }

}

private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    switch (e.KeyCode)
    {
case Keys.Space:
    if (m_isRunning == true)
timer1.Stop();
    else
timer1.Start();
    m_isRunning = !m_isRunning;
    break;
default:
    break;
    }
}

private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    timer1.Stop();
    m_isRunning = false;
    m_iterator = listBox1.SelectedIndex;
    pictureBox1.Image = Bitmap.FromFile(m_filenameList[m_iterator]);
    pictureBox1.Update();
}
```

2.3 Affichage dans 2 fenêtres extérieures

Ajouter au projet une seconde form (projet → ajouter un formulaire Windows → selectionner Form). Il s'agit d'une nouvelle classe, que vous nommerez *FormDisplay*. Sur *FormDisplay*, placez un contrôle de type PictureBox, mettez l'attribut Dock de cette PictureBox à Fill, afin que cette pictureBox occupe tout l'espace de la form, **et modifiez l'accessibilité de cette PictureBox, en mettant la propriété Modifiers à Public**.



La zone d'affichage PictureBox de la Form1 disparaitra, il restera les boutons et la ListBox.
Déclarez dans le code de la classe Form1 deux instances de FormDisplay (variables membres) :

```
public FormDisplay m_FormA = new FormDisplay();
public FormDisplay m_FormB = new FormDisplay();
```

On utilisera maintenant les Forms m_FormA et m_FormB pour afficher les images, en alternance. Pour afficher une image dans m_FormA, il suffira d'écrire : `m_FormA.pictureBox1.Image = Bitmap.FromFile(...);`. La m_formA peut être rendue visible avec `m_FormA.Show();` invisible avec `m_FormA.Hide();` et mise au premier plan avec `m_FormA.Select();` Pour distinguer entre la fenêtre m_formA et la fenetre m_formB, on peut changer leur titre :

```
m_FormA.Text = "Form A";
m_FormB.Text = "Form B";
```

Modifiez votre code (essentiellement dans la méthode `timer1_Tick()`) de la classe Form1 pour obtenir le fonctionnement suivant :

La première image sera chargé dans m_FormA, qui sera placé à l'avant plan. Lorsque la durée d'affichage de la 1ere image expirera, l'image suivante sera placée dans m_FormB, qui passera au premier plan, et ainsi de suite de façon à toujours alterner entre m_FormA et m_FormB.
Ce n'est pas si simple, et il est recommandé de ne pas s'embarquer dans des algorithmes complexes, avec des opérateurs "modulo" ou des index et itérateurs multiples. Le plus simple est d'associer une donnée membre à la classe DisplayForm, de type Boolean `isOnTop`. La fenêtre qui sera sur le dessus aura cette donnée membre `isOnTop = true`, et un test avec `isOnTop` servira pour savoir dans quelle Form charger l'image suivante.

Elements de correction :

```
private void timer1_Tick(object sender, EventArgs e)
{
    if (m_filenameList.Count > 0)
    {

        if (m_FormA.isOnTop == true)  // c'est la formA qui est au dessus
        {
            m_FormB.pictureBox1.Image = Bitmap.FromFile(m_filenameList[m_iterator]);
            m_FormB.pictureBox1.Update();
            m_FormA.isOnTop = false;
            m_FormB.isOnTop = true;
            m_FormB.Select();
        }
        else // c'est donc FormB qui est au dessus
        {
            m_FormA.pictureBox1.Image = Bitmap.FromFile(m_filenameList[m_iterator]);
            m_FormA.pictureBox1.Update();
            m_FormA.isOnTop = true;
```

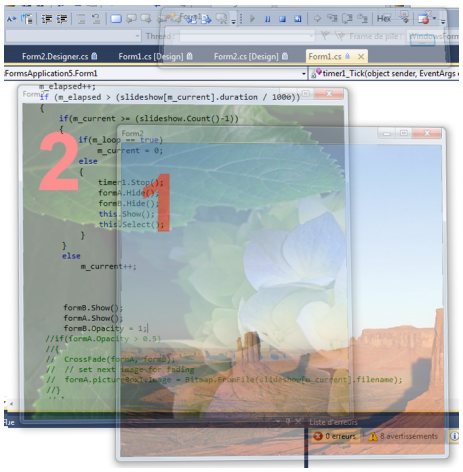
```
                m_FormB.isOnTop = false;
                m_FormA.Select();
            }
            m_iterator++;
            if (m_iterator >= m_filenameList.Count)
                m_iterator = 0;
        }

    }
```

Montrez ce fonctionnement en alternance, en ouvrant d'abord un répertoire nommé TestPair contenant un nombre pair d'images (4 images) puis un répertoire nommé TestImpair contenant et un nombre impair d'images (5 images par exemple).

2.4 Exploitation de la transparence des contrôles

Pour pouvoir passer au fondu enchainé, il faut légèrement modifier le fonctionnement : au lieu de charger l'image et de l'afficher de suite, l'image sera chargée dans une FormDisplay qui sera maintenue invisible. Lorsque le timer s'écoule, la FormDisplay invisible s'affichera progressivement (en passant son opacité de 0 à 100%), l'ancienne, qui se retrouve en dessous, deviendra invisible (opacité = 0%) et sera chargée avec la nouvelle image, puis mise sur le dessus. **Il y a ainsi toujours une image chargée d'avance par rapport à l'image affichée.**
La méthode permettant le fondu est la suivante (ce n'est pas la peine de recopier les commentaires ...). Vous remarquerez que la valeur de l'opacité varie entre 0 et 1 :

```
public void CrossFade(Form FromForm, Form ToForm)
{
  ToForm.Select();            // ToForm est mis au premier plan, on suppose ToForm comme transparent
                              // c'est à dire avec Opacity=0
  while (ToForm.Opacity < 1)  // dans cette boucle, on va rendre ToForm progressivement opaque
  {
    ToForm.Opacity += 0.01;
    System.Threading.Thread.Sleep(10);
  }
  FromForm.Opacity = 0;       // FromForm était au second plan. Il est rendu transparent en prévision
}                             // du prochain changement
```



Utilisez cette méthode lorsque vous souhaitez passer de la m_FormA à la m_FormB, puis de la m_FormB à la m_FormA, etc... Cette méthode remplace les appels à `Hide()` et `Show()` ou `Select()` que vous avez probablement utilisés à l'étape précédente.
Ne passez à la suite que si vous êtes satisfait de votre fonctionnement.

3 Diaporama en mode plein-écran

Avant de passer en mode plein écran, on souhaite pouvoir quitter le diaporama lorsqu'on appuie sur une touche, n'importe laquelle. Lors du diaporama, c'est soit m_FormA soit m_FormB qui seront à l'avant plan (et m_FormA comme m_FormB sont des instances de FormDisplay). Il faut donc ajouter à la classe FormDisplay un gestionnaire en réaction à l'événement KeyDown ou PreviewKeyDown (valider aussi la propriété PreviewKey = true). L'instruction pour quitter l'application est `Application.Exit();`
Assurez-vous que vous pouvez quitter l'application. Ensuite, ajoutez un test pour ne quitter l'application que si l'utilisateur frappe la touche ESC (`Keys.escape`)
Après avoir vérifié que l'on peut quitter l'application, on peut enfin passer au plein écran... Ce n'est qu'une question de réglage des propriétés de la classe FormDisplay :

1. Enlever la bordure de type dialogue : positionnez la valeur de FormBorderStyle sur None
2. Sélectionner du noir comme couleur de fond (BackColor)
3. Mettre la boite de dialogue en plein écran : WindowState sur Maximize
4. Empêcher l'affichage de chaque boite de dialogue dans la barre des tâches de Windows : ShowInTaskbar sur False
5. Et pour le contrôle pictureBox qui se trouve sur FormDisplay : SizeMode sur Center (ou Stretch, ou Zoom, cela dépend de la taille des images et sur la façon dont vous souhaitez les afficher)

Testez votre diaporama, puis "soignez" votre condition de départ (Que se passe t'il au lancement de l'application ? Comment afficher la première image ? éventuellement avec un fondu depuis une surface noire). Montrez le fonctionnement final.

4 Réglages supplémentaires possibles au clavier

4.1 Modification de la vitesse d'affichage des images

– Durée d'affichage de chaque image plus longue si appui sur la touche [+] du pavé numérique (Keys.Add) (ajouter 0.5 sec à chaque action sur cette touche)
– Durée d'affichage de chaque image plus courte si appui sur la touche [-] du pave numérique (Keys.Subtract)
Comme les frappes clavier sont détectées par des instances de la classe FormDisplay, il faut pouvoir accéder au Timer, qui est une donnée de Form1 (mettre le timer comme donnée publique ou alors écrire une ou des méthode(s) publique(s) de la Form1 permettant le réglage de l'intervalle du timer).

4.2 Défilement aléatoire des images

Si appui sur la touche [r] (pour random), l'ordre d'affichage devient aléatoire. Il faut quand même que toutes les images soient affichées une fois avant de boucler. Le moyen le plus simple pour faire cela est de dupliquer la liste contenant le nom des images a afficher (attention a réellement copier la liste), et on travaille uniquement sur la copie de la liste. Dans le fonctionnement standard, on affiche les images en prenant la liste dans l'ordre, et on retire de la liste l'image que l'on vient d'afficher. Dans le mode aléatoire, on récupère une valeur aléatoire entre 1 et `List<>.count`, on affiche l'image désignée par ce tirage aléatoire et on retire celle-ci de la liste. D**ans les deux cas, lorsque la liste est à 0, il faut a nouveau demander une copie de la liste d'origine.**

5 Pour ceux qui ont fini

Implémentez une animation type "Ken Burns Effect", c'est à dire en ajoutant de très léger zoom et de très légère rotation lors de l'affichage des images.
Le moyen le plus simple est d'ajouter un second timer, avec un intervalle plus court (par exemple 200 millisecondes). A chaque Tick de ce second timer, on applique une déformation à l'image. La rotation a été abordée en TP, le Zoom suit la même logique :

```
Graphics gfx = Graphics.FromImage(m_FormA.pictureBox1.Image);
gfx.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;

// on positionne le centre du zoom au centre de l'image
gfx.TranslateTransform((float)m_FormA.pictureBox1.Width / 2, (float)m_FormA.pictureBox1.Height / 2);
// on applique le zoom (si les valeurs sont superieures à 1, sinon on dezomme)
gfx.ScaleTransform(1.005F, 1.005F);
// on redessine l'image à la bonne position (ne pas oublier que l'origine a été déplacé au centre du PictureBox)
gfx.DrawImage(m_FormA.pictureBox1.Image, -m_FormA.pictureBox1.Width / 2, -m_FormA.pictureBox1.Height / 2);
// ne pas oublier !!!
m_FormA.pictureBox1.Invalidate();
```

Un bon "Ken Burns Effect" peut être obtenu si pour chaque image, de façon aléatoire, le centre du zoom est placé soit en haut a gauche, soit en haut à droite, soit au centre.
Les images du dossier applescreensaver.zip sont idéales pour expérimenter ce mode de fonctionnement.