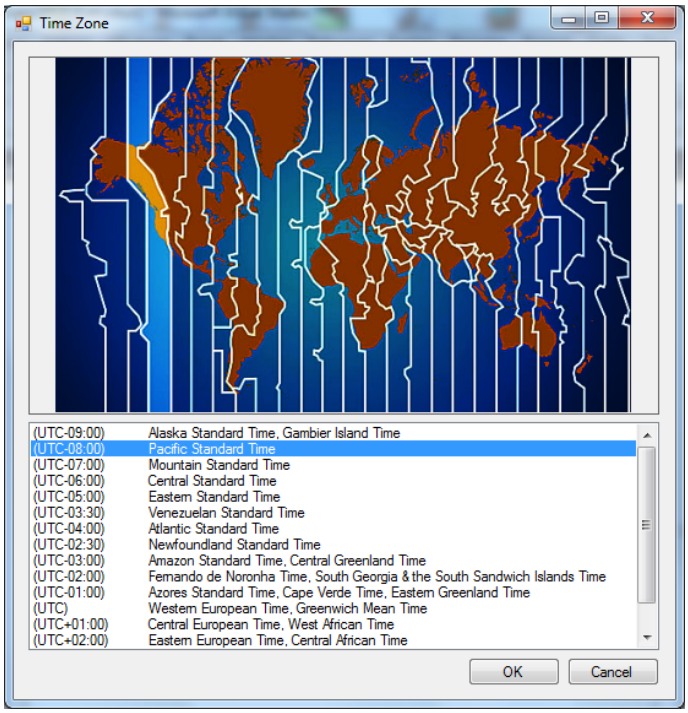
Université de la Rochelle
INFO-12605C - IHM - TD9

© B. Besserer, R. Péteri

Année universitaire 2016-2017

1 Utilisation d'une ListBox



On souhaite réaliser une boite de dialogue permettant le type d'affichage présenté ci-contre (sélection d'une zone temporelle avec affichage de la mappemonde). La boite de dialogue en soit est assez simple : un contrôle de type PictureBox nommé pictureBox1, un contrôle de type ListBox, nommé listBox1, et deux boutons.

1. Commençons par la ListBox. Chaque article d'une ListBox est un objet, nous allons donc définir une classe d'objet `myTimeZoneDescription` comportant un nom (String) et un objet de classe TimeSpan.
   Écrivez cette classe. Elle doit disposer d'un constructeur acceptant le nom de la zone et une valeur comme paramètres, une surcharge de la méthode ToString permettant d'afficher la valeur correctement formatée, puis le nom de la zone séparé par une tabulation `'\t'`, et enfin un accesseur de type *property* pour la valeur (lecture seule) Attention au formatage : le décalage temporel étant signé ou nul, l'affichage doit en tenir compte (affichage des symboles). La propriété `TimeSpan::Ticks` donne une valeur signée ou nulle et peut être utilisée comme test ; et l'utilisation du formatage `TimeSpan::ToString(@"hh\:mm")` permet l'affichage des heures et minutes seuls (en non les jours, secondes, etc...)
   (une quinzaine de ligne de code).
   Pour information, voici les constructeurs possible pour la classe TimeSpan :

Name	Description
TimeSpan(Int64)	Initializes a new instance of the TimeSpan structure to the specified number of ticks.
TimeSpan(Int32, Int32, Int32)	Initializes a new instance of the TimeSpan structure to a specified number of hours, minutes, and seconds.
TimeSpan(Int32, Int32, Int32, Int32)	Initializes a new instance of the TimeSpan structure to a specified number of days, hours, minutes, and seconds.
TimeSpan(Int32, Int32, Int32, Int32, Int32)	Initializes a new instance of the TimeSpan strrructure to a specified number of days, hours, minutes, seconds, and milliseconds.

Eléments de correction :

```
public class myTimeZoneDescription
{
```

```
        private String m_Location;
        private TimeSpan m_Offset;
        public TimeSpan GetOffset
        {
            get { return m_Offset; }
        }

        // constructeur
        public myTimeZoneDescription(String loc, int hoffset, int moffset)
        {
            m_Location = loc;
            m_Offset = new TimeSpan(hoffset, moffset, 0);
        }

        // surcharge de ToString()
        public override String ToString()
        {
            if (m_Offset.Ticks < 0)
                return ("(UTC-" + m_Offset.ToString(@"hh\:mm") + ')' + '\t' + m_Location);
            else if (m_Offset.Ticks > 0)
                return ("(UTC+" + m_Offset.ToString(@"hh\:mm") + ")" + '\t' + m_Location);
            else
                return ("(UTC)" + '\t' +  '\t' + m_Location);
        }
    }
```

2. En supposant qu'un programme de test de type Windows Forms fait office de boite de dialogue, vos éléments (image, listbox) doivent être initialisés au lancement du programme. **Ecrivez les lignes de code correspondant au remplissage de 4 premières lignes de la ListBox - voir copie d'écran pour le contenu de ces 4 premières lignes.**

```
namespace Exam2013
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            pictureBox1.Image = new Bitmap("timezone_none.png");
            ... ajouter le remplissage du contrôle ListBox1
        }
    }
}
```

Eléments de correction :

```
namespace Exam2013
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.listBox1.Items.Add(new myTimeZoneDescription("Alaska Standard Time, Gambier Island Time"
            this.listBox1.Items.Add(new myTimeZoneDescription("Pacific Standard Time", -8, 0));
            this.listBox1.Items.Add(new myTimeZoneDescription("Mountain Standard Time", -7, 0));
            this.listBox1.Items.Add(new myTimeZoneDescription("Central Standard Time", -6, 0));
            ...
        }
    }
}
```

3. Lors de la création de la boite de dialogue, on suppose qu'aucune timezone n'est sélectionnée et que la mappemonde ne montre aucune zone en surbrillance. L'affichage de cette mappemonde **neutre** est effectué dans le constructeur de la boite de dialogue (voir code précédent). Lorsque l'utilisateur clique sur une *timezone* de la liste, il faut charger l'image avec la zone temporelle correspondante en surbrillance. On suppose qu'un ensemble de X images soient disponibles, autant d'images que de timezone, avec pour chaque image une timezone particulière en surbrillance.

   Sans en écrire le code, proposez un moyen de charger la bonne image lors du clic, en sachant que la liste n'est pas toujours dans l'ordre présenté (elle peut être triée différemment)... comment faire la relation entre element de la liste et image à afficher !

Eléments de correction :

– *Soit un schéma de nommage des fichiers selon l'écart au zero UTC, dans ce cas, il vous faudra créer le nom du fichier a partir des informations du TimeSpan*

– *Soit l'extension de la classe* `myTimeZoneDescription` *des objets contenu dans la listbox, en ajoutant une donnée membre supplémentaire correspondant au nom du fichier.*

Idéalement, mettre un try/catch lors du chargement des images (il peut y avoir un fichier manquant)

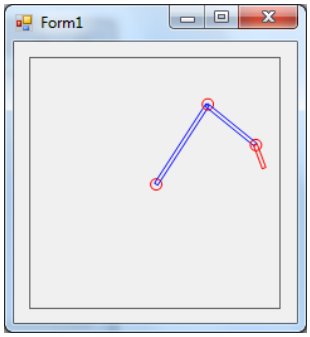
Par exemple ici, on construit un nom de fichier nommé par exemple `"timezone_-01_30_00.png"` *(on utilise donc essentiellement le décalage horaire pour former le nom du fichier, mais il faut se debarasser de caractères inadmissibles pour des noms de fichier, par exemple on remplace " :" par "_".*

```
myTimeZoneDescription toto  = listBoxZones.Items[listBoxZones.SelectedIndex] as myTimeZoneDescription;
TimeSpan t = toto.GetOffset;
String s = t.ToString();
s = s.Replace(':', '_');
s = "timezone_" + s + ".png";
try
{
pictureBox1.Image = new Bitmap(s);
}
catch
{}
```

4. Sans en écrire le code, décrivez comment vous pourriez implémenter la fonctionnalité suivante : L'utilisateur clique dans la mappemonde et la timezone correspondante est sélectionnée dans la liste. Nommez les difficultés que vous pourriez rencontrer et des propositions pour les résoudre.

Eléments de correction : *- Zone non rectangulaire, nécessité d'avoir une description vectorielle de timezone. Beaucoup plus facile en WPF.*

2 Transformations géométriques sur des images - suite au prochain TD



On souhaite dessiner un bras de robot avec une interaction à la souris (sélection des segments du bras et modification de l'angle de rotation de chaque bras). Le bras du robot est composé de 3 segments nommés UpperArm, LowerArm et Wrist, les angles de rotation étant nommés angle1 (entre la base et UpperArm), angle2 (entre UpperArm et LowerArm) et angle3 (entre LowerArm et Wrist). La base est le centre de la surface de dessin (un panel nommé PicCanvas).
Effectuer l'ensemble des tracés à partir de l'origine (0,0) de la form ou du panel serait assez complexe. Certes, le tracé de la première articulation est simple :

```
// tracé de la première articulation, qui est centrée dans la zone de dessin (panel)
gr.DrawEllipse(Pens.Red, (picCanvas.ClientSize.Width / 2)-4,
                                    (picCanvas.ClientSize.Height / 2) -4, 9, 9);
```

Mais cela se complique dès le tracé du premier segment (UpperArm) si celui-ci n'est pas strictement vertical ou horizontal ; la méthode `DrawRectangle` ne convient plus, il faut utiliser `DrawPolygon`, gérer 4 points par segment du robot, et évidemment cela devient de plus en plus complexe.
Le but d'un framework étant de soulager le travail du développeur, le contexte graphique de l'environnement .NET permet donc de manipuler des matrices de transformation. Plus précisement, le contexte graphique permet de mémoriser une matrice de transformation dite matrice globale (telle que toute opération de dessin subit l'influence de cette matrice).
The Graphics class provides several methods for building up a composite world transformation : MultiplyTransform, RotateTransform, ScaleTransform, and TranslateTransform. For Example :

```
myGraphics.ScaleTransform(1, 0.5f);
myGraphics.TranslateTransform(50, 0, MatrixOrder.Append);
myGraphics.RotateTransform(30, MatrixOrder.Append);
myGraphics.DrawEllipse(myPen, 0, 0, 100, 50);
```

Attention : les combinaisons de matrices (= mutiplication) ne sont pas commutatives. Le résultat d'une mise à l'échelle (scale) suivi d'une translation puis d'une rotation ne donne pas le même résultat qu'une translation, une rotation suivi d'une mise à l'échelle. D'où le second paramètre qui peut être `MatrixOrder.Append` (la matrice est combiné en fin de chaîne) ou `MatrixOrder.Prepend` (la transformation est placée en début de chaîne).
A savoir qu'une classe Matrix existe dans le namespace System.Drawing.Drawing2D. Pour les objets de cette class, les opérations Translate, Rotate, Multiply, Reset et Invert sont définies.
Voici le début de la méthode permettant de tracer le bras du robot :

```
public Rectangle UpperArm = new Rectangle(0, -1, 75, 3);
public Rectangle LowerArm = new Rectangle(0, -1, 50, 3);
public Rectangle Wrist = new Rectangle(0, -1, 20, 3);
public Rectangle Connexion = new Rectangle(-4, -4, 9, 9);

private void DrawRobotArm(Graphics gr)
{
  // Translate to center of form.
  float cx = picCanvas.ClientSize.Width / 2;
  float cy = picCanvas.ClientSize.Height / 2;
  gr.TranslateTransform(cx, cy);

  // Draw the shoulder centered at the origin.
  gr.DrawEllipse(Pens.Green, Connexion);

  // Rotate at the shoulder.
  // (Negative to make the angle increase counter-clockwise).
  gr.RotateTransform(angle1, MatrixOrder.Prepend);

  // Draw the first arm.
  gr.DrawRectangle(Pens.Blue, UpperArm);
  ...
```

1. Où aurait été tracé le rectangle correspondant au premier segment du bras du robot si, dans le code ci-dessus, l'opération aurait été `gr.RotateTransform(angle1, MatrixOrder.Append);`

   Eléments de correction : *Concernant les transformations globales : Soit vous imaginez le tracé réalisé et ensuite on applique les transformations dans l'ordre. La rotation est effectué en utilisant 0,0 comme centre.*

2. Complétez lez code pour obtenir le tracé complet du bras du robot.

   Eléments de correction : *Par exemple...*

```
private void DrawRobotArm(Graphics gr)
{
    const int UpperArmLength = 75;
    const int LowerArmLength = 50;
    const int WristLength = 20;

    gr.SmoothingMode = SmoothingMode.AntiAlias;
    gr.Clear(picCanvas.BackColor);

    // For each stage in the arm, draw and then *prepend* the
    // new transformation to represent the next arm in the sequence.

    // Translate to center of form.
    float cx = picCanvas.ClientSize.Width / 2;
    float cy = picCanvas.ClientSize.Height / 2;
    gr.TranslateTransform(cx, cy);

    // Draw the shoulder centered at the origin.
    gr.DrawEllipse(Pens.Red, -4, -4, 9, 9);

    // Rotate at the shoulder.
    // (Negative to make the angle increase counter-clockwise).
    gr.RotateTransform(-scrJoint1.Value, MatrixOrder.Prepend);

    // Draw the first arm.
    gr.DrawRectangle(Pens.Blue, 0, -1, UpperArmLength, 3);

    // Translate to the end of the first arm.
    gr.TranslateTransform(UpperArmLength, 0, MatrixOrder.Prepend);

    // Draw the elbow.
    gr.DrawEllipse(Pens.Red, -4, -4, 9, 9);

    // Rotate at the elbow.
    gr.RotateTransform(-scrJoint2.Value, MatrixOrder.Prepend);

    // Draw the second arm.
    gr.DrawRectangle(Pens.Blue, 0, -1, LowerArmLength, 3);

    // Translate to the end of the second arm.
    gr.TranslateTransform(LowerArmLength, 0, MatrixOrder.Prepend);

    // Draw the wrist.
    gr.DrawEllipse(Pens.Red, -4, -4, 9, 9);

    // Rotate at the wrist.
    gr.RotateTransform(-scrJoint3.Value, MatrixOrder.Prepend);

    // Draw the third arm.
    gr.DrawRectangle(Pens.Blue, 0, -1, WristLength, 3);
}
```

3. On souhaite maintenant pouvoir sélectionner à la souris l'un des 3 segments du bras du robot ; le segment sélectionné sera dessiné dans une autre couleur. Pour savoir si un point se trouve dans un rectangle, il existe une méthode de la classe `Rectangle` nommée `Contains`. Pour pouvoir utiliser cette méthode dans notre cas, ils faut faire subir aux coordonnées du clic les transformations inverses de celles subies lors du tracé. Pour cela, on utilisera trois instances de la classe Matrix : matrix1, matrix2 et matrix3. Ces matrices mémorisent l'inverse des transformations qu'ont subis respectivement les segments UpperArm, LowerArm et Wrist. Complétez le code de DrawRobotArm pour mémoriser ces matrices, et écrivez la méthode qui sera associée à la méthode deleguée PicCanvas.Click permettant de savoir si le clic a été fait dans l'un ou l'autre des segments.
4. Après avoir sélectionné un segment du bras, l'utilisateur peut déplacer celui-ci en rotation à l'aide de la molette de la souris. A savoir que MouseWheel n'apparait pas dans la liste des événements, il faut donc associer votre méthode à la méthode déléguée picCanvas.MouseWheel "à la main" :
   `this.picCanvas.MouseWheel += new MouseEventHandler(picCanvas_MouseWheel);`
   Comme toujours, la signature de cette méthode sera :
   `private void picCanvas_MouseWheel(object sender, System.Windows.Forms.MouseEventArgs e)`
   et l'argument `MouseEventArgs a` contient un champ `Delta` qui correspond à la rotation de la molette (valeur signée). Écrivez le code pour pouvoir tourner le segment sélectionné.

Eléments de correction :

```
private void picCanvas_Click(object sender, EventArgs e)
 {
     picCanvas.Focus();
     Point[] loc = new Point[1];
     loc[0] = ((MouseEventArgs)e).Location;
     matrix1.TransformPoints(loc);
     if (UpperArm.Contains(loc[0]))
     {
         UpperArmPen = Pens.Red;
         LowerArmPen = WristPen = Pens.Blue;
         picCanvas.Refresh();
         return;
     }
     loc[0] = ((MouseEventArgs)e).Location;
     matrix2.TransformPoints(loc);
     if (LowerArm.Contains(loc[0]))
     {
         LowerArmPen = Pens.Red;
         UpperArmPen = WristPen = Pens.Blue;
         picCanvas.Refresh();
         return;
     }
     loc[0] = ((MouseEventArgs)e).Location;
     matrix3.TransformPoints(loc);
     if (Wrist.Contains(loc[0]))
     {
         WristPen = Pens.Red;
         UpperArmPen = LowerArmPen =Pens.Blue;
         picCanvas.Refresh();
         return;
     }
 }
```