

(www.unvrai.com, www.unfaux.com)

Introduction

ZedGraph est une librairie destinée aux développements .NET aussi bien en WinForm qu'en WebForm. Les classes qu'elle fournit sont extrêmement souples et paramétrables. Il est possible de représenter et de configurer quasiment n'importe quel type de graphe, d'un histogramme à la représentation de fonctions mathématiques

L'utilisation de cette librairie est simplifiée par l'adoption d'un grand nombre de valeurs par défaut pour l'aspect des graphiques.

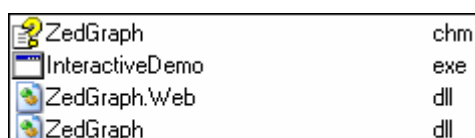
ZedGraph est un développement open-source que l'on peut trouver sur SourceForge (<http://sourceforge.net/projects/zedgraph>), accompagné d'exemples et de documentation. La dernière version 5.0 requière .NET 2.0.

Remarque:

Dans ce document nous nous occupons uniquement de la partie WinForm de ZedGraph.

Installation

Après avoir téléchargé ZedGraph, en plus des sources et autres fichiers, intéressons-nous au minimum vital, à savoir les 4 fichiers suivants:



ZedGraph	chm
InteractiveDemo	exe
ZedGraph.Web	dll
ZedGraph	dll

Le premier est une aide pour les développeurs

Le second est une application exemple, montrant les diverses possibilités offerte par ZedGraph.

Le troisième est le seul fichier incontournable permettant de développer des applications de type WinForm incluant ZedGraph.

Le quatrième est l'équivalent pour le développement d'application WebForm.

Utilisation

ZedGraph est accessible en tant que composant dans la boîte à outils de Visual Studio. Mais pour cela, il faut:

- cliquer avec le bouton droit de la souris par exemple dans le groupe Général de la boîte à outils
- choisir "Choose items...", puis "Browse..." dans la fenêtre qui apparaît
- naviguer jusqu'au dossier dans lequel se trouve le fichier **ZedGraph.dll** et valider le choix

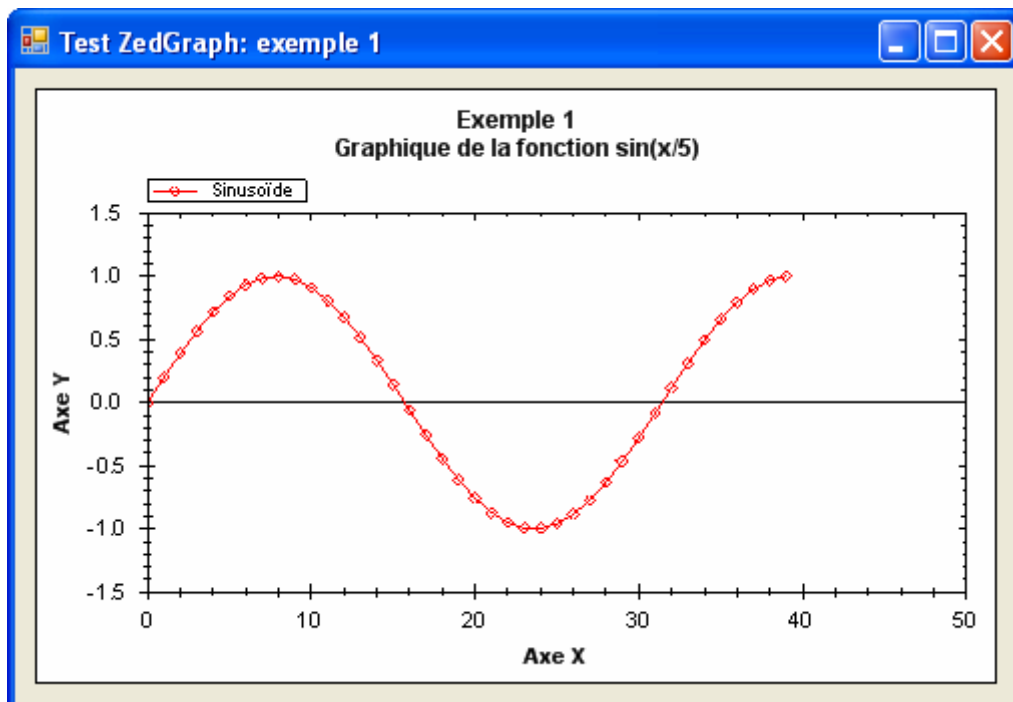
A partir de ce moment le composant ZedGraph doit apparaître sous la forme suivante dans la boîte à outils de Visual Studio:



Maintenant le composant est utilisable dans n'importe quelle application WinForm. Nous allons voir deux exemples basés sur ZedGraph.

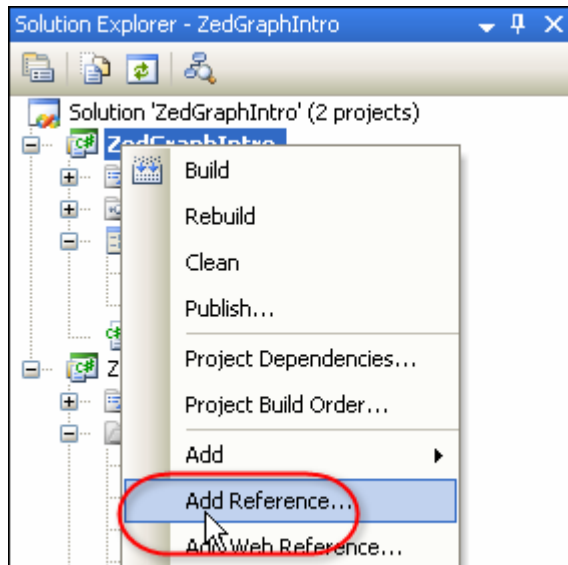
Exemple 1: une simple application

Dans cet exemple nous allons construire, pas à pas, une application très simple utilisant ZedGraph et traçant le graphique d'une fonction sinusoïdale. Voici l'aspect final de ce programme:



Pour atteindre ce résultat il faut effectuer les étapes suivantes:

- créer un nouveau projet de type "Application Windows" dans Visual Studio
- une fois que la fenêtre de l'application apparaît, cliquer avec le bouton droit sur le projet et choisir "Add Reference..."



- choisir "Browse" et sélectionner le fichier **ZedGraph.dll**
- sélectionner le composant ZedGraph dans la boîte à outils et le placer sur la fenêtre
- le faire arriver près des bords de la fenêtre
- l'événement FormLoad appelle CreerGraphique(zg1) qui effectue l'affichage du graphique, et est défini comme suit:

```
private void Form1_Load(object sender, EventArgs e)
{
    CreerGraphique(zg1); // zg1 est le nom du composant ZedGraphControl
}
```

- voici les instructions permettant de paramétrer et d'afficher le graphique:

```
private void CreerGraphique(ZedGraphControl zgc)
{
    // référence vers le "canevas"
    GraphPane Pane = zgc.GraphPane;

    // Les titres
    Pane.Title.Text = "Exemple 1\n Graphique de la fonction sin(x/5)";
    Pane.XAxis.Title.Text = "Axe X";
    Pane.YAxis.Title.Text = "Axe Y";

    // Quelques points pour la fonction Sinus
    double x, y1;
    PointPairList list1 = new PointPairList();
    for (int i = 0; i < 40; i++)
    {
        x = (double)i; // valeur sur l'axe X
        y1 = Math.Sin((double)i * 0.2); // valeur sur l'axe Y
        list1.Add(x, y1); // ajout du point à la liste
    }
}
```

```

// Génération d'une courbe sinus
LineItem Courbel = Pane.AddCurve("Sinusoïde",list1, Color.Red, SymbolType.Diamond);
zgc.AxisChange();
}

```

- afin de disposer d'un graphique dont les dimensions varient en fonction du redimensionnement de la fenêtre, voici le code lié à l'événement Resize:

```

private void Form1_Resize(object sender, EventArgs e)
{
    SetSize();
}

private void SetSize()
{
    // permet de laisser une marge de 10 pixels tout autour du graphique
    zgl.Location = new Point(10, 10);
    zgl.Size = new Size(ClientRectangle.Width - 20, ClientRectangle.Height - 20);
}

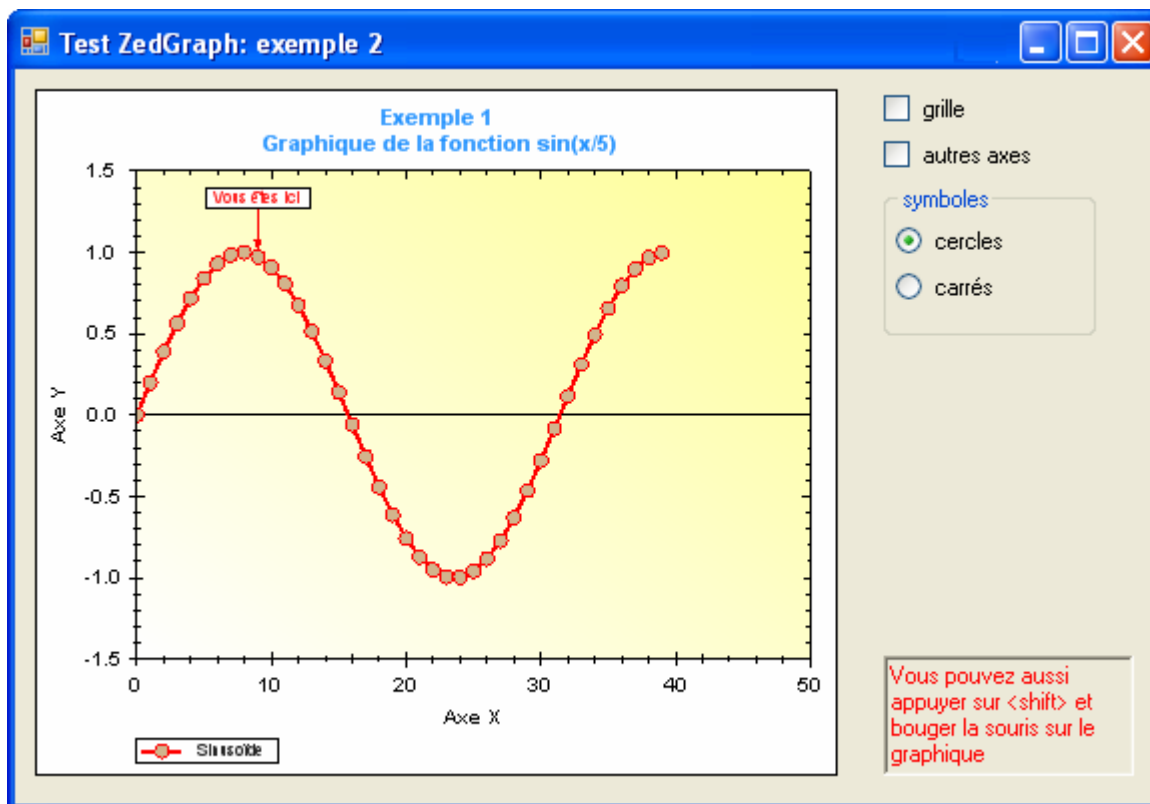
```

- ne pas oublier de préciser:

```
using ZedGraph;
```

Exemple 2: améliorations

Dans ce deuxième exemple, nous avons repris l'application précédente et nous avons ajoutés quelques fonctionnalités et/ou paramétrages. Voici comment elle se présente:



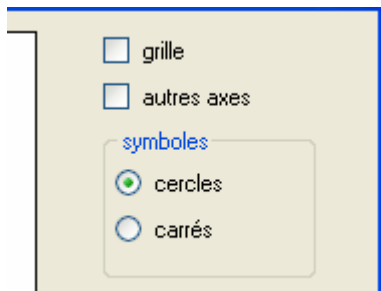
Nous allons passer en revue ces modifications et adjonctions:

- changement de la couleur du titre

Exemple 1 Graphique de la fonction $\sin(x/5)$

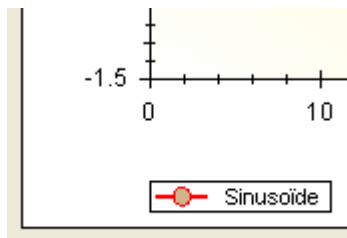
```
Pane.Title.FontSpec.FontColor = Color.DodgerBlue;
```

- agrandissement de la marge à droite du graphique pour laisser la place aux options:



```
private void SetSize()
{
    // permet de laisser une marge de 10 pixels tout autour du graphique
    zgl.Location = new Point(10, 10);
    zgl.Size = new Size(ClientRectangle.Width - 180, ClientRectangle.Height - 20);
}
```

- position de la légende en bas

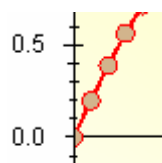


```
Pane.Legend.Position = ZedGraph.LegendPos.Bottom;
```

- Ajout d'un fond dégradé du blanc au jaune (inclinaison 45 degrés):

```
Pane.Chart.Fill = new Fill(Color.White, Color.FromArgb(255, 255, 150), -45F);
```

- symboles plus grands et remplis, ainsi que l'épaisseur du tracé qui passé de 1 à 2 pixels



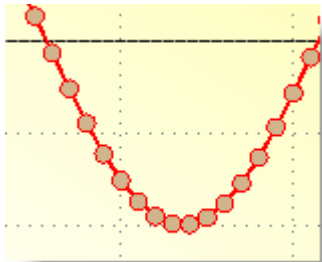
```
Courbel.Symbol.Size = 10.0F; // symboles plus grands
Courbel.Symbol.Fill = new Fill(Color.Tan); // couleur de remplissage des symboles
Courbel.Line.Width = 2.0F; // épaisseur de la courbe
```

- Ajout d'un texte "Vous êtes ici" encadré et accompagné d'une flèche rouge:



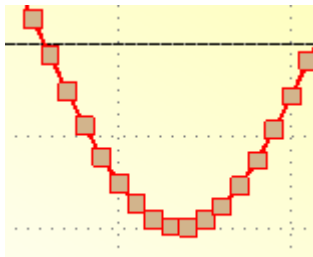
```
TextObj monText = new TextObj("Vous êtes ici", 9F, 1.4F);
monText.FontSpec.FontColor = Color.Red;
monText.Location.AlignH = AlignH.Center;
monText.Location.AlignV = AlignV.Top;
Pane.GraphObjList.Add(monText);
ArrowObj maFleche = new ArrowObj(Color.Red, 10F, 9F, 1.4F, 9F, 1.0F);
Pane.GraphObjList.Add(maFleche);
```

- case à cocher permettant d'afficher une grille pointillée de couleur grise:



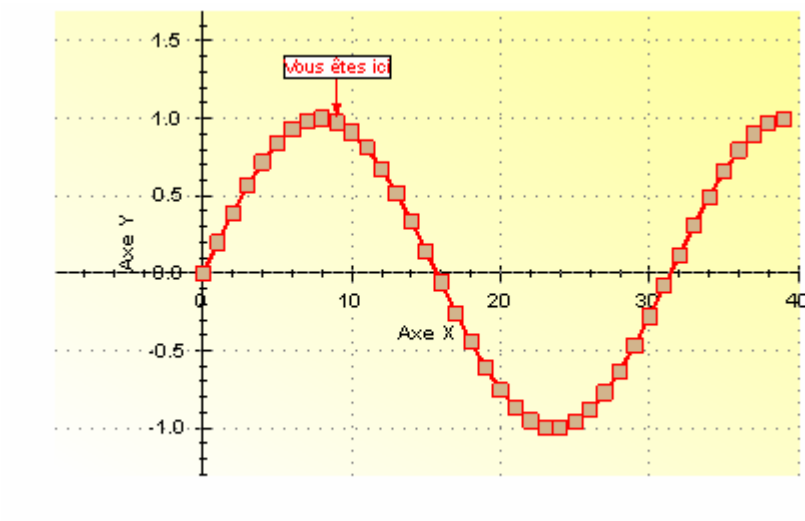
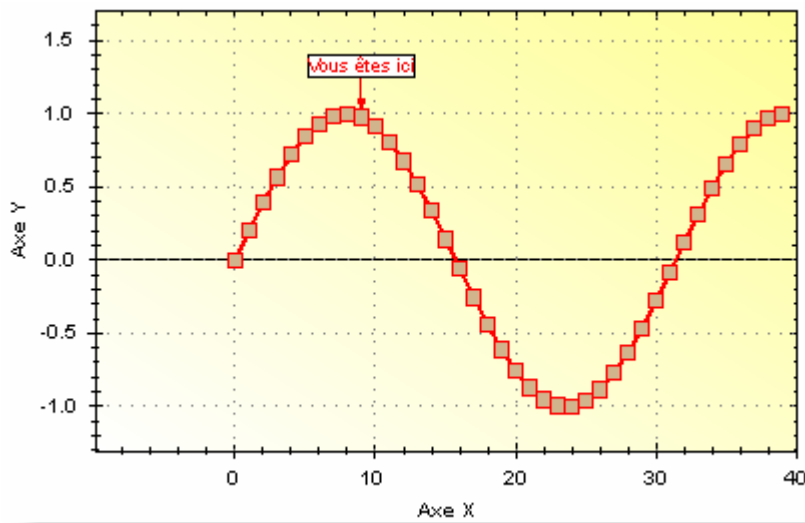
```
private void cbGrille_CheckedChanged(object sender, EventArgs e)
{
    // référence vers le "canevas"
    GraphPane Pane = zgl.GraphPane;
    // Ajout d'une grille
    Pane.XAxis.MajorGrid.IsVisible = cbGrille.Checked;
    Pane.YAxis.MajorGrid.IsVisible = cbGrille.Checked;
    Pane.XAxis.MajorGrid.Color = Color.Gray;
    Pane.YAxis.MajorGrid.Color = Color.Gray;
    zgl.Refresh();
}
```

- possibilité d'afficher des carrés au lieu des cercles pour les symboles:



```
private void rbCercle_CheckedChanged(object sender, EventArgs e)
{
    // référence vers le "canevas"
    GraphPane Pane = zgl.GraphPane;
    LineItem Courbel = (LineItem)(Pane.CurveList["Sinusoïde"]); // on récupère la courbe
    if (rbCarre.Checked)
        Courbel.Symbol.Type = SymbolType.Square; // choix des symboles
    if (rbCercle.Checked)
        Courbel.Symbol.Type = SymbolType.Circle;
    zgl.Refresh();
}
```

- possibilité d'avoir une autre représentation des axes. Voici les deux possibilités après avoir fait défiler légèrement le graphique (d'abord la situation par défaut, puis la situation modifiée):



```
private void cbAxes_CheckedChanged(object sender, EventArgs e)
{
    // référence vers le "canevas"
    GraphPane Pane = zgl.GraphPane;
    if (cbAxes.Checked)
    {
        Pane.YAxis.Cross = 0.0; // L'axe Y coupe l'axe X en 0.0
        Pane.XAxis.Cross = 0.0; // L'axe X coupe l'axe Y en 0.0
    }
    else
    {
        Pane.YAxis.CrossAuto = true;
        Pane.XAxis.CrossAuto = true;
    }
    // Suppression du cadre et des graduations en haut à droite
    Pane.Chart.Border.IsVisible = !cbAxes.Checked;
    Pane.XAxis.MajorTic.IsOpposite = !cbAxes.Checked;
    Pane.XAxis.MinorTic.IsOpposite = !cbAxes.Checked;
    Pane.YAxis.MajorTic.IsOpposite = !cbAxes.Checked;
    Pane.YAxis.MinorTic.IsOpposite = !cbAxes.Checked;
    zgl.Refresh();
}
```

Conclusion

Comme on peut le voir au travers de ces deux exemples, l'appropriation de ce composant ne pose pas de difficulté particulière. Il est, bien entendu, possible de modifier une quantité incroyable de paramètres, ou même d'avoir une famille de graphiques. Bonne exploration.