TP C#0 : Ceci n'est que le commencement ...

1 Cultivez vous qu'ils disaient ...

1.1 Le C# , la plateforme .NET, Microsoft

Le C# est un langage de programmation créé en 2001 par la société Microsoft et notamment par l'un de ses employés, Anders Hejlsberg [1]. Il est orienté objet et fortement typé, c'est-à-dire qu'il garantit que les types de données employés décrivent correctement les données manipulées. Sa syntaxe est relativement semblable à d'autres langages tels que le C++ et le C, et reste également très proche du Java dont il reprend la syntaxe mais aussi certains concepts.

Le C# a été créé afin que la plate-forme Microsoft .NET soit dotée d'un langage permettant d'utiliser toutes ces capacités. Microsoft .NET (prononcé "Dot Net") est le nom donné à un ensemble de produits et de technologies informatiques de Microsoft, dont le but est de rendre les applications facilement portables sur Internet, on parle aussi de Framework .NET.

Vous pourrez noter également que des implémentations libres du C# et de sa plate-forme d'exécution existent, comme le projet Mono maintenu à l'heure actuelle par la société Xamarin depuis 2011, ou bien encore comme dotGNU maintenu actuellement par la Free Software Foundation. L'idée fondatrice de ces projets est qu'une application en C# puisse être developpée et s'exécuter sans modification sur une plate-forme propriétaire comme Windows ou libre comme Linux.

1.2 Visual Studio

Microsoft Visual Studio est un IDE (Integrated Development Environment) qui contient une suite de logiciels de développement pour Windows conçue par Microsoft. Il s'agit d'un ensemble complet d'outils de développement permettant de générer des applications, dont notamment un *Garbage Collector* permettant une meilleure gestion de la mémoire ou encore de *IntelliSense* une des meilleures implémentations d'auto-complétion connue jusqu'à ce jour.
Tous ces outils permettent de développer des applications dans divers langages s'appuyant sur le Framework .NET comme évidemment le C# mais aussi comme le F#, le Visual C++ et le Visual Basic. Il vous est donc possible de développer des types d'application très variés : applications graphiques, applications en console, des bibliothèques de classes, des services Windows ou encore des sites web.

Vous pouvez donc dès à présent utiliser Visual Studio 2010 disponible sur vos racks, ou bien le retrouver sur MSDNAA avec votre compte EPITA, ou encore via Visual Studio Express que vous pouvez télécharger gratuitement sur les sites de Microsoft.

Le Caml c'est fini, place au C# maintenant !

1. `http://fr.wikipedia.org/wiki/Anders_Hejlsberg`

2 Archive et rendu

Qui dit Tp, dit rendu et qui dit rendu dit archive (correcte) de rendu.
Voici l'architecture de rendu à suivre : (en remplaçant évidemment "login_x" par votre login, à moins que vous ne vous appeliez Xavier Loginard)

```
rendu-tp-login_x.zip
     | login_x/
         |_ AUTHORS
         | HelloWorld/
              | HelloWorld.sln
              | HelloWolrd/
         | MyCrypt/
              | MyCrypt.sln
              | MyCrypt/
         | MyTinyDemineur/
              | MyTinyDemineur.sln
              | MyTinyDemineur/
         | MyMorpion/
              | MyMorpion.sln
              | MyMorpion
```

3 Quelques bases avant de bien commencer

Jusqu'à maintenant la plupart d'entre vous n'avez fait que du Caml en utilisant sa syntaxe et son vocabulaire.
Vous allez maintenant découvrir un autre langage, on le présente plus, et oui c'est bien le C# .

Mais avant de commencer, nous allons vous faire un petit topo. Premièrement il vous faut comprendre qu'en C# chaque ligne de code exécutée est une instruction se terminant par un point virgule ';'.
Toutes les instructions sont contenues entre des accolades {}.
Vous pouvez commenter votre code en utilisant un double slash devant // ou bien en encadrant votre code ainsi /* ... votre code ...*/.
C# définit les types suivants : int, float, double, bool, char, string, object, etc.
Il est possible d'effectuer des conversions entre tous ces types, pour cela vous pouvez utiliser des méthodes dédiées et présentes de base dans C# ou le transtypage[2].

Voici quelques exemples :

```
char c = 'Z';
int i = 42;

// Voici un commentaire !
/*
Ou même tout un bloc.
*/
string the_Answer = i.ToString();
```

En C# , les fonctions s'écrivent de la manière suivante :

2. Un problème, un souci, une chose pas claire ? n'hésitez pas à poser une question !

```
type_de_retour nom_de_la_fonction(type1 variable1, type2 variable2)
{
        // Code me a river ...
}
```

Vous pouvez aussi créer des blocs conditionnels, comme en Caml :

```
if (Condition)
{
}
else
{
}
```

Mais cela est juste à titre indicatif, vous le verrez en temps voulu dans les autres TP de C# , dans lesquels vous découvrirez d'ailleurs qu'il existe d'autres mots clés, permettant entre autres de faire des boucles ou encore de déclarer des structures. Vous pouvez bien sûr déjà les utiliser si vous en êtes capable, mais attendez vous à devoir en expliquer le principe rapidement devant vos ACDC, et pourquoi pas devant toute la classe.

4 Hello world !

4.1 Nouveau projet

Vous vous en doutiez peut être, on ne commence pas un nouveau langage de programmation sans apprendre à dire bonjour via le très célèbre " Hello world !".

Premièrement, ouvrez Visual Studio 2010, si besoin dites lui que vous voulez travailler avec du C# et créez un nouveau projet nommé HelloWorld :

Fichier => Nouveau => Projet => Application Windows Forms.

Visual Studio affiche alors la fenêtre principale de développement de votre projet. De nombreux menus sont placés entre vos mains pour que vous paramétriez et utilisiez de manière ergonomique et optimale votre nouvel outil : Visual Studio 2010. Vous êtes totalement libre en ce qui concerne la configuration de votre IDE, vous pouvez même le faire en rose avec une jolie police, cela ne nous regarde pas ! Mais sachez juste qu'il existe de nombreux outils intégrés pour la création de vos projets et que vous pouvez éventuellement ajouter des extensions à Visual Studio pour parfaire votre interface de travail pour le reste de l'année.

Nous attirons votre attention également sur une option très pratique si vous êtes de nature écologiste ou bien tout simplement si vous ne voulez pas que vos yeux finissent brûlés après des heures de code acharnées et ceci grâce au superbe fond blanc de Visual Studio. Vous pouvez par exemple suivre les instructions suivantes :

Outils => Options => Environnement => Polices et couleurs => Premier plan => Blanc
Outils => Options => Environnement => Polices et couleurs => Arrière plan => Noir

Voilà une bonne chose de faite, vous ne trouvez pas ?
Et bien maintenant il est grand temps de se mettre à bosser !

La suite du programme se trouve dans le volet *Explorateur de solutions*, affichez le et regardez ce que l'on trouve à l'intérieur :

- Solution 'HelloWorld' (1 projet)
  - HelloWorld
    - Properties
      - *Ressources spécifiques au projet. (Images, ...)*
    - References
      - *Liste des bibliothèques inclues dans le programme.*
    - Form1.cs : **Votre code !**
      - Form.Designer.cs : *Code généré par l'éditeur graphique, ne pas modifier.*
      - Form1.resx :*Ressources de la Form.*
      - Program.cs :*Point d'entrée du programme.*

Un simple double clic sur l'un de ces éléments ouvrira l'éditeur ou le visualiseur adéquat.

Vous avez tout compris ?
Visual Studio n'a plus aucun secret pour vous ?
Et bien il est temps de faire vos preuves dans ce cas, en attaquant de suite l'exercice 1 !

5 Exercice 1 :

5.1 My_Hello_World

Pour cet exercice, vous allez utiliser des Windows Forms pour implémenter un "Hello world".

Les Windows Forms sont des applications utilisant .NET pour créer une interface graphique, avec des boutons, du texte, des checkbox, ...

Pour cela reprenez votre projet créé tout à l'heure, vous avez à votre disposition une fenêtre que vous pouvez redimensionner, renommer, remplir d'éléments issus de la Toolsbox (Boîte à Outils). Cette dernière contient tous, et je dis bien tous, les éléments qui vous sont nécessaires et que vous pouvez utiliser pour faire des interfaces graphiques et plus particulièrement celles des exercices du TP.

Chaque élément que vous ajoutez à la fenêtre de votre programme possède des attributs, que vous pouvez modifier à volonté dans les propriétés de l'élément, tels que le nom de l'élément, le texte qu'il contient, sa couleur, sa propriété d'être cliquable ou non ... Pour obtenir les propriétés d'un objet ? Rien de plus simple : *Clic droit sur l'élément => Propriétés*

Mais comment fait-on pour associer des actions aux boutons ? C'est là que le code entre en jeu (enfin !). En double-cliquant sur un élément, une fonction est automatiquement générée.
Pour modifier un élément, vous devez récupérer son identifiant.
Il est visible dans ses attributs. Par exemple, pour changer le texte d'un bouton, il suffit d'écrire dans la fonction :

```
button1.Text = "Test";
```

En remplaçant button1 par l'identifiant de votre bouton.

Passons maintenant à notre Hello World. Le Windows Forms possédera plusieurs options permettant de le personnaliser.
Le Hello World sera affiché en tant que texte du bouton.
Le "Hello" sera affichable dans plusieurs langues différentes, suivi d'un choix entre deux noms personnalisés.

Il sera également possible de modifier la couleur du texte.
Mais un schéma vaut mieux qu'un long discours.

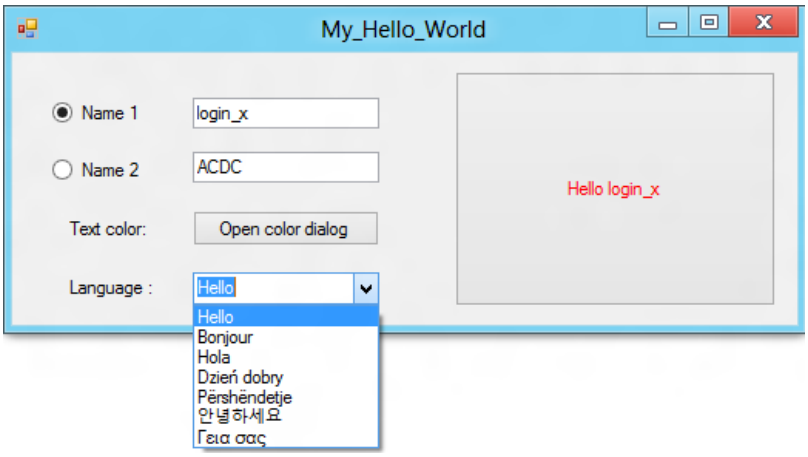

FIGURE 1 – Hello world.

6 Exercice 2 :

6.1 My_Crypt

Pour vous initier très légèrement à la cryptographie, vous aller devoir coder un programme qui remplira les fonctions suivantes :

- Cacher le texte que vous avez tapé par des *,
- Ré-afficher le texte d'origine après modification,
- Effectuer une rotation de votre choix sur le texte entré.

Mais avant le code il vous faut l'interface, qu'elle soit simple ou compliquée. L'interface basique que vous pouvez utiliser se résume à la suivante : une TextBox, trois Boutons.

Libre à vous une fois encore de faire parler votre imagination pour le reste.

Voici une représentation de ce qu'on peut attendre de votre imagination :

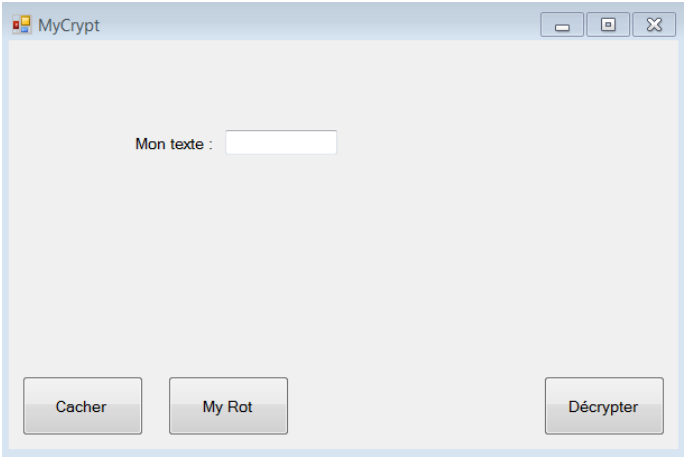

FIGURE 2 – MyCrypt.

7 Exercice 3 :

7.1 My_Tiny_Demineur

7.1.1 Version basique

Après 2 mois d' électronique, vous vous sentez fin prêt à mettre en pratique vos cours pour faire un peu de déminage. Mais avant de vous lancer dans un champ de mine, il faut tester vos compétences avec une simulation ultra-réaliste.

Vous allez donc devoir réaliser un Démineur simplifié.

Le principe est simple. Une bombe sera dissimulée parmi quatre boutons aléatoirement.
Pour remporter la partie, il faut cliquer sur les trois boutons qui ne possèdent pas la bombe.

La fenêtre sera composée de cinq boutons :
- un qui servira à relancer une partie,
- quatre autres pour le désamorçage.

Le texte du bouton principal changera en fonction de l'état de la partie :
- si une partie est en cours, il affichera un "-",
- si le joueur a gagné, il affichera un " :)",
- sinon " :(".

Les boutons de jeu ne devront être cliquables qu'une seule fois par partie, (indice : Pour désactiver un bouton il faut qu'il ne soit pas Enabled). Ainsi, au moment de relancer la partie il faudra penser à les réactiver.

En C# , les classes sont des ensembles permettant de contenir un groupe de variables et fonctions. Toutes les variables que vous définissez à l'intérieur peuvent, et doivent être visible uniquement à l'intérieur de cette classe.

Comme le C# est un langage fortement orienté objet, toutes vos fonctions devront être dans ces classes (mais vous pouvez avoir plusieurs classes). Il est recommandé d'utiliser une variable contenant la position de la bombe, que vous devrez déclarer dans votre Classe si vous voulez que chaque fonction puisse l'utiliser.

Elle pourra contenir des valeurs de 0 à 3 correspondant à la case contenant la bombe.

Vous devez déjà connaître comment définir un nombre aléatoire en Caml, voici la syntaxe en C# :

```
/* Déclaration d'un random et initialisation */
Random rand = new Random();

/* Renvoi un nombre aléatoire entre 0 et 42 exclu */
bomb = rand.Next(0, 42);
```

Résultat final :

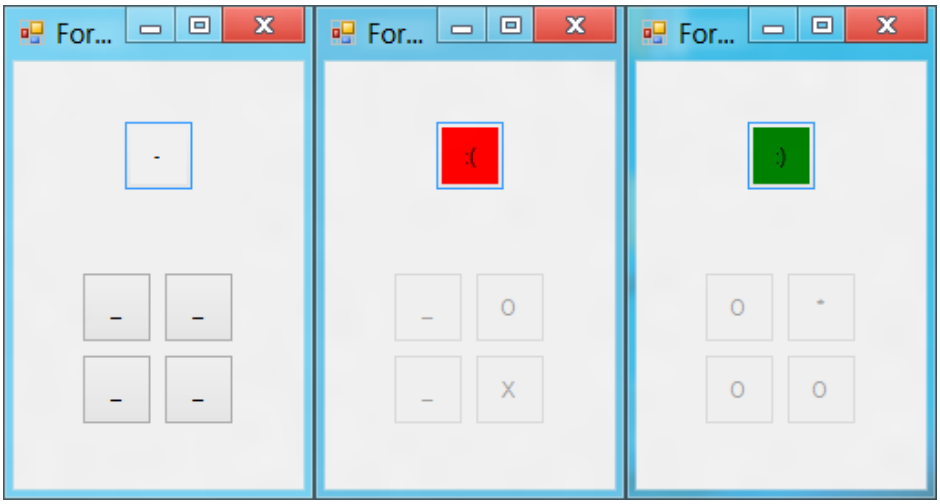

FIGURE 3 – My Tiny Demineur.

7.1.2 Bonus

Maintenant que votre démineur fonctionne, pourquoi ne pas l'améliorer ?
Voici quelques idées de bonus :

- Ajouter des couleurs sur les boutons,
- Afficher les statistiques Victoire/Défaite de la session courante,
- Un démineur en 3D avec XNA.

8 Bonus !

Tout le TP était trop facile pour vous ?
Aucun problème, car quand y en a plus et bah y en a encore !

8.1 Le Morpion

Pour changer de contexte pédagogique, vous allez coder un Morpion (OXO) [3]. Ce jeu n'est pas sensé vous être inconnu, quoi qu'il en soit, pour ceux qui n'ont jamais connu les longues parties de Morpion les matins d'hivers pendant un cours peu intéressant, ils trouveront sans doute leur bonheur sur internet.

Pour les autres, nous allons voir un peu plus en détails ce que l'on appelle un Morpion ou plutôt communément appelé Tic-Tac-Toe.

Pour faire un bon Morpion vous aurez besoin :

- une grille de neuf cases,
- deux joueurs au minimum, vous pouvez gérer plus de joueurs mais ça devient un compliqué,
- deux "pions" différentiables, au hasard nous prendrons un X et un O,
- un cerveau (oui oui je vous assure c'est parfois utile) pour mettre tout cela en forme dans Visual studio,
- pour une partie gagnée il faut que l'un des deux joueurs aligne trois pions sur une ligne (verticale, horizontale, diagonale).

3. `http://fr.wikipedia.org/wiki/Tic-tac-toe_(jeu)`

FIGURE 4 – Une partie gagnée.

FIGURE 5 – Une partie gagnée.

Pour vous aider voici quelques conseils :
- On utilisera un bouton carré pour représenter une case, si la case a déjà été jouée on désactivera le bouton.
- Pour un soucis de priorité, les X commencent, ce qui veut dire que la première case est remplie par un X, la deuxième par un O, la troisième par un X, etc.
- Pour la détection de victoire, nous vous rappelons que vous connaissez la valeur de chaque bouton, il vous reste juste à regarder les boutons autour.

Voici un petit exemple de ce que nous demandons :

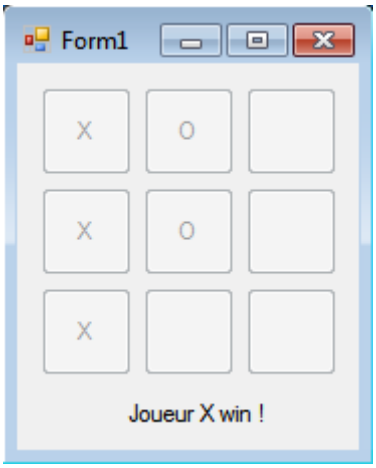

FIGURE 6 – Une partie gagnée.

9 Conclusion

In ACDC You Trust.