Utilisation d'une manette de jeu avec Visual Studio en C#

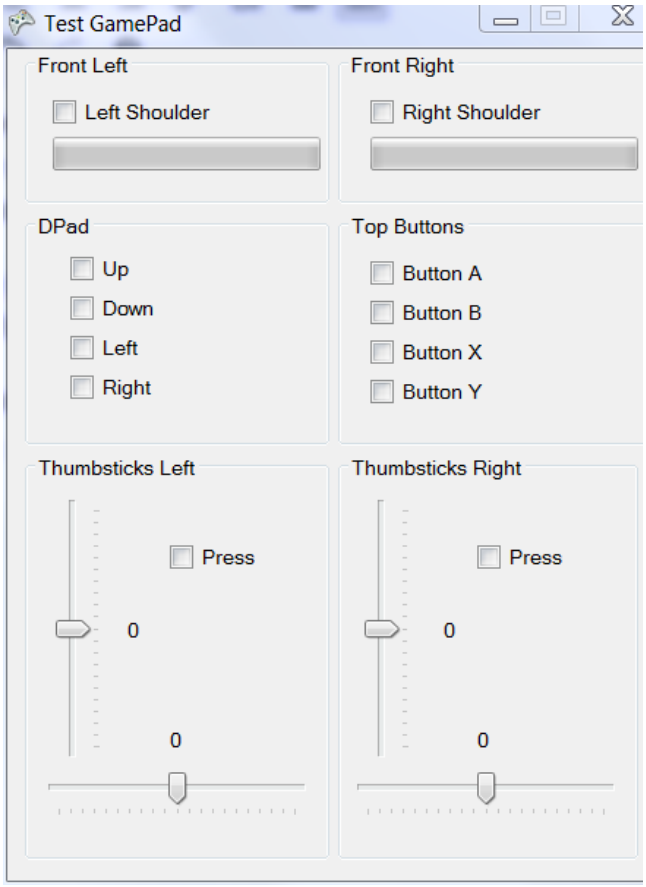
Posted on 21 janvier 2017

Le contrôle d'objets mobiles à distance nécessite l'emploi d'une télécommande ou d'une manette. De nombreux shields Arduino existent sur le marché et permettent de satisfaire à ce besoin (ex : http://www.robotshop.com/ca/en/dfrobot-joystick-input-shield-arduino.html). Cependant, la première idée qui vient souvent à l'esprit des élèves est d'utiliser leur vielle manette de jeu Xbox, PS, ou autre compatible.

Leur utilisation dans le cadre d'un projet de développement nécessite l'utilisation de librairies spécifiques (python : pygame, Microsoft : DirectX, SlimDX, XNA, ...). Bien qu'il y ait de nombreux tutoriels sur le net, je n'ai pas trouvé de solution simple, à la porté d'élèves de terminal STI2D novices en programmation.

Toutefois, l'environnement XNA Game Studio met à disposition du développeur un framework qui contient les outils nécessaires à la gestion des manettes connectées à un PC sous Windows. Il peut être utilisé sous Visual Studio dans le cadre d'un projet Windows Form classique.

Je vous propose de réaliser une application de test des principaux contrôles possibles avec une manette compatible Logitech.



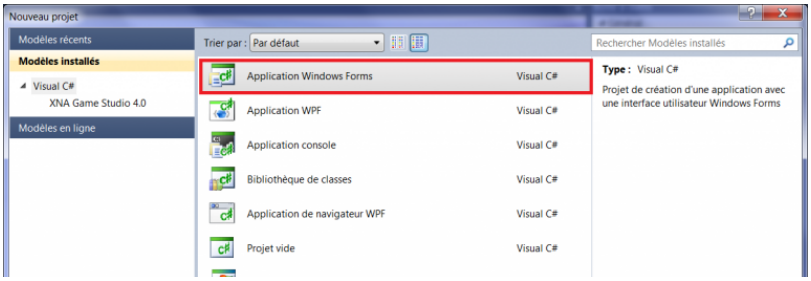
Installation de XNA Game Studio

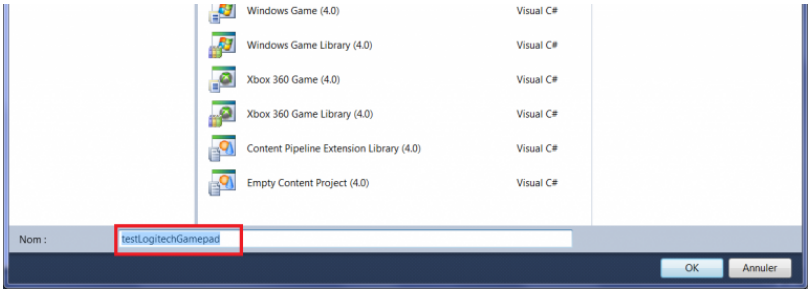
Téléchargez et installez XNA Game Studio 4.0 : https://www.microsoft.com/en-us/download/details.aspx?id=23714

Nouveau projet Visual Studio

Ouvrez Visual Studio et démarrez un nouveau projet Application Windows Form. Remarquez au passage l'ajout de nouveau types de projets réalisables : XNA Game Studio 4.0 (Windows ou Xbox 360 games ou libraries).

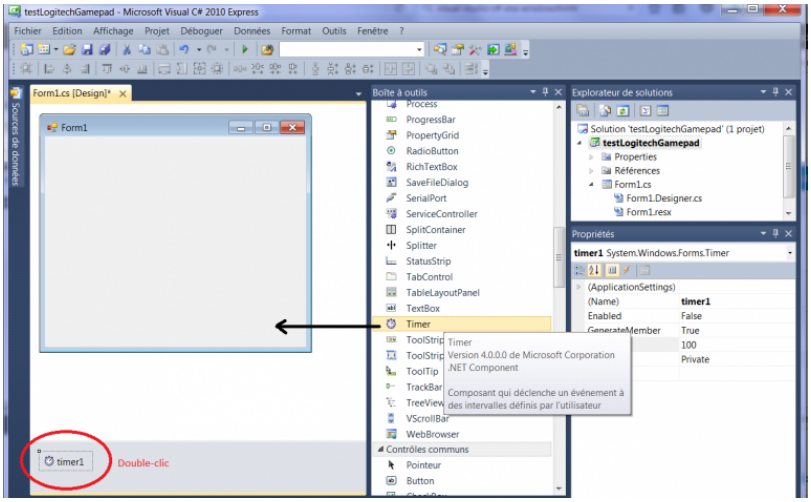
Nommez le projet *testLogitechGamepad*





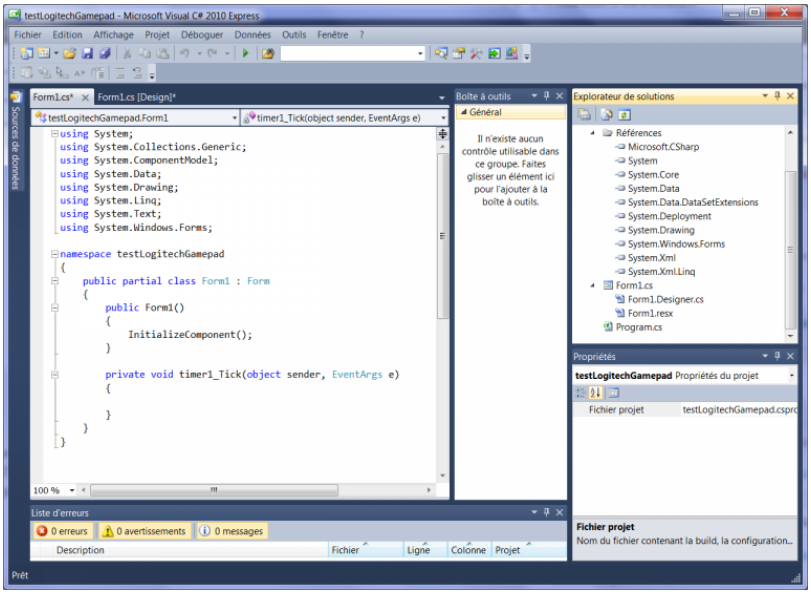
Insertion du timer

Les contrôles de la manette seront scrutés à intervalles réguliers à l'aide d'un timer. Dans la boite à outils, cliquez sur le timer et glissez le sur le formulaire Form1.



Double-cliquez sur l'icône du timer dans le Designer pour créer l'évènement ***Timer1_Tick()*** qui sera exécuter à chaque échéance du timer séparées par l'intervalle définit dans les propriétés (100 ms par défaut). Remarquez également la propriété ***Enable*** à ***false*** qu'il faut évidemment passer à ***true*** pour que le timer démarre dès l'exécution de l'application.

Le code de l'application ressemble actuellement à ceci :

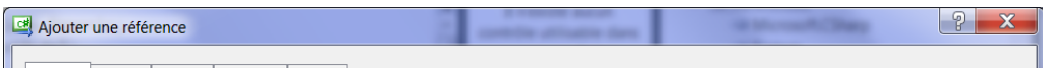


A ce stade, il n'est pas encore possible de lire les contrôles de la manette. Il nous faut ajouter le framework XNA à notre projet.

Ajout du framework XNA

Dans l'explorateur de solutions, cliquez droit sur ***Références***, puis sélectionnez ***Ajouter une référence***.

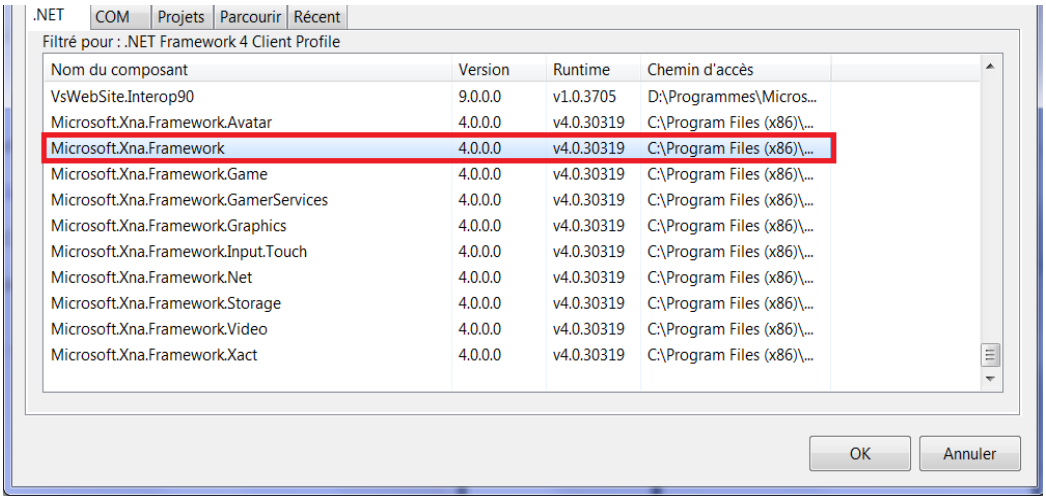
Dans l'onglet .NET, choisissez Microsoft.Xna.Framework et validez. La référence est ajoutée au projet.



Catégories

Liste de Liens

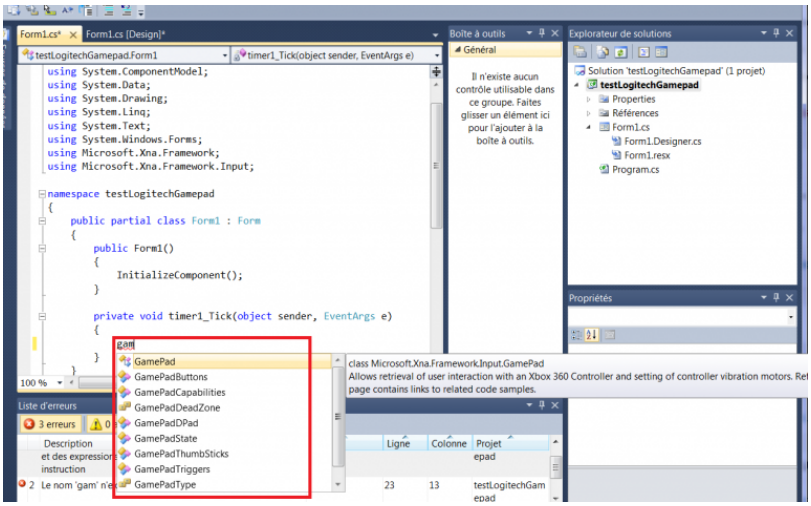
Archive



L'accès aux attributs et aux méthodes des objets de ce framework ne seront utilisables qu'après avoir indiqué l'utilisation des espaces de noms appropriés.

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Input;
```

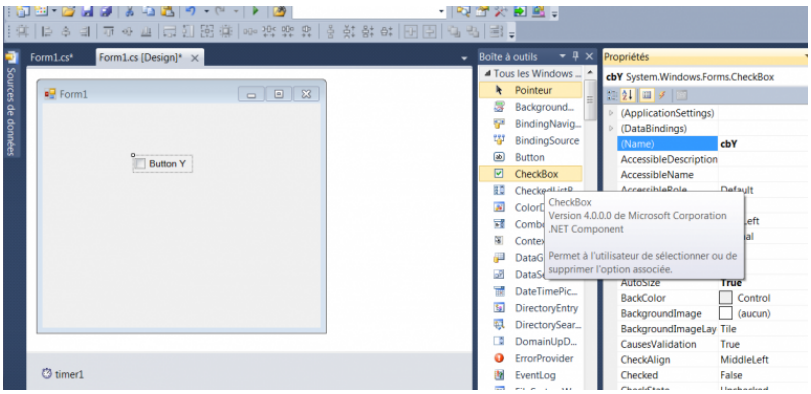
Désormais, vous pouvez accéder à votre manette ! Faites le test : dans le code, saisissez ***game*** l'autocomplétion vous propose la suite...



Tester un bouton

Je vous propose de tester l'appui sur le bouton **Y** à droite de la manette :

- placez sur le formulaire une case à cocher (CheckBox),
- modifiez sa propriété ***Name*** : ***cbY***
- modifiez sa propriété ***Text*** : ***Button Y***



avril 2018
mars 2018
février 2018
juin 2017
mars 2017
février 2017
janvier 2017
avril 2016
mars 2016
décembre 2015
octobre 2014
janvier 2014
novembre 2013
mai 2013
avril 2013
mars 2013
février 2013
janvier 2013
décembre 2012
novembre 2012
juin 2012
mars 2012
octobre 2011
février 2011
décembre 2010

Meta

Connexion
RSS
Commentaires RSS

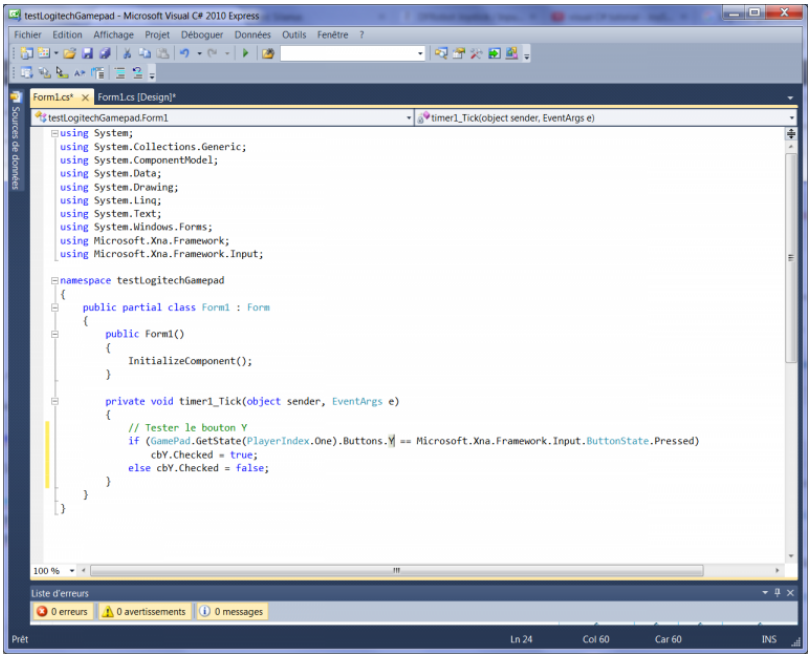
Dans le code, on va tester l'état du bouton **Y** puis mettre à jour la propriété ***Checked*** du composant ***cbY*** :

- Dans l'éditeur de code, placez vous dans la description de la méthode :

```
private void timer1_Tick(object sender, EventArgs e)
```

- Testez l'état du bouton Y de la manette du joueur 1 et mettre à jour l'état de la CheckBox ***cbY*** :

```
if(GamePad.GetState(PlayerIndex.One).Buttons.A == Microsoft.Xna.Framework.Input.ButtonState.Pressed)
     cbY.Checked = true;
else cbY.Checked = false;
```

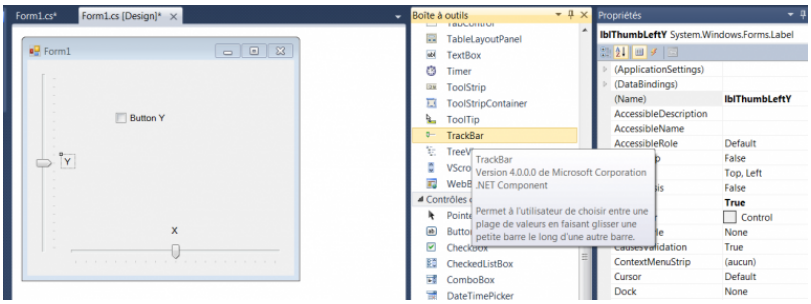


Tester un joystick

Les joysticks sont appelés ThumbSticks Left et Right. Ils délivrent les nombres X et Y compris entre -1 et 1, la position de repos étant 0.

Nous allons placer sur le formulaire 2 TrackBars et deux Labels pour visualiser les position en X et Y du joystick ThumbSticks Left (donc celui de gauche).

- Insérez un composant TrackBar sur le formulaire :
  - ***Name : tbThumbLeftX***
  - ***Orientation : Horizontal***
  - ***Minimum : -1000***
  - ***Maximum : 1000***
  - ***TickFrequency : 100***
  - ***Value : 0***
- Insérez un nouveau composant TrackBar sur le formulaire :
  - ***Name : tbThumbLeftY***
  - ***Orientation : Vertical***
  - ***Minimum : -1000***
  - ***Maximum : 1000***
  - ***TickFrequency : 100***
  - ***Value : 0***
- Insérez un label au dessus du curseur du TrackBar ***tbThumbLeftX***
  - ***Name : lblThumbLeftX***
  - ***Text : X***
- Insérez un label à côté du curseur du TrackBar ***tbThumbLeftY***
  - ***Name : lblThumbLeftY***
  - ***Text : Y***

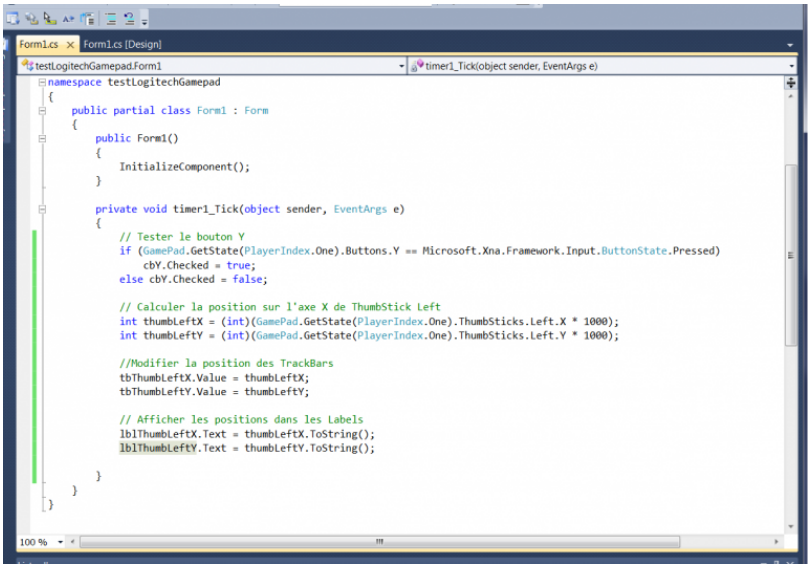


Dans le code, on va calculer les positions renvoyées par le joystick sur les axes X et Y entre -1000 et 1000, modifier en conséquence la position des TrackBars et afficher leurs positions dans les Labels :

```
// Calculer la position sur l'axe X de ThumbStick Left
int thumbLeftX = (int)(GamePad.GetState(PlayerIndex.One).ThumbSticks.Left.X * 1000);
int thumbLeftY = (int)(GamePad.GetState(PlayerIndex.One).ThumbSticks.Left.Y * 1000);

//Modifier la position des TrackBars
tbThumbLeftX.Value = thumbLeftX;
tbThumbLeftY.Value = thumbLeftY;

// Afficher les positions dans les Labels
lblThumbLeftX.Text = thumbLeftX.ToString();
lblThumbLeftY.Text = thumbLeftY.ToString();
```



A vous de jouer ... !

Télécharger le projet

Remplis sous: Divers, OS, Visual Studio C#

Commentaires

Commentaires (0) Trackbacks (0) (Souscrire aux commentaires de cet article)

Désolé, le formulaire de commentaire est fermé pour le moment

Azolla : le plus court chemin entre deux points ... »

« La météo dans ma ville

Aller en haut ↑