LES SEMI CROUSTILLANTS

Norme de Codage

Appliquée Aux Projet Annuel

Jonathan Bihet

20/02/2013

Document présentant la norme de codage pour le projet annuel de 3A DJV.

Contenu

Qu'est-ce Que C'est ?

- La norme de codage s'applique aux scripts C# du projet annuel, sur Unity 3D, des étudiants de 3A DJV de l'ESGI.
- Elle concerne :
    - La dénomination.
    - La présentation globale (paragraphes).
    - La présentation locale (lignes).
    - Les headers.
    - La documentation et les commentaires.

Présentation Générale

- Une ligne ne doit pas excéder 80 colonnes.
- Une seule instruction par ligne.
- Une fonction ne doit, de préférence, pas excéder 25 lignes entre les accolades.

Les Commentaires & La Documentation

- Les commentaires dans le corps des méthodes ne seront autorisés que dans les callback de MonoBehaviour.
- Commentaire simple, //, seront les seuls autorisés dans les corps de méthodes.
- Commentaire multi ligne, /**/, seront utilisé principalement/uniquement pour les headers.
- Pour la documentation on utilisera la méthode habituelle du C# à savoir /// pour définir la zone de documentation.
- <summary></summary> pour décrire l'élément souhaité.
- <param></param> pour décrire les paramètres de l'élément.
- <returns>< /returns> pour décrire ce que retourne l'élément.
- <value>< /value> permet de décrire la valeur de retour ou d'entrée d'une propriété. Je ne pense pas qu'on l'utilise beaucoup mais on ne sait jamais.
- <paramref>< /paramref> permet d'indiquer que le mot dans le commentaire est un paramètre de la fonction.
- <c>< /c> Le tag "c" permet d'indiquer que le mot est un mot faisant partie du code.
- <remarks></remarks> permet de mettre des remarques sur une classe.
- <exception>< /exception> permet d'informer sur le(s) type(s) d'exception(s) que la fonction peut lever. Vous devez donner dans le tag, la condition pour que l'exception soit déclenchée. La propriété cref du tag permet de spécifier le type d'exception que vous documentez.
- Le tag "example" permet de spécifier le texte accompagnant un exemple d'utilisation d'une méthode ou d'une classe.
- le tag "code" permet de marquer le texte qui suit comme étant du code.
- Le tag "see" permet de créer un lien vers un élément documenté grâce au nom de cet élément. Il doit être placé à l'intérieur d'un tag "summary" ou "remarks".
- Le tag "seealso" permet de créer un lien en bas de la page de documentation. Il est utilisé pour renvoyer le lecteur à un complément d'information.
- Pour les callback de MonoBehaviour (Start, Update, FixedUpdate…), la doc est quasi-inutile, on privilégiera les commentaires pertinents.

```
/// <summary>
///  This is a test class.
///  with a method <c>myMethod</c> to do everything when used in <c>Update</c>.
///  The Test class have two args, <c>_foo</c> & <c>_bar</c>.
/// </summary>
/// <remarks>
///   Secret info : this class is useless.
/// </remarks>
public class Test : MonoBehaviour
{
    public Transform _foo;
    public float _bar;

    /// <summary>
    /// Start is called just before any of the Update methods is called
    /// the first time.
    /// </summary>
    void Start()
    {
        //we do nothing.
        _foo = this.transform;
        bar = _foo.x;

    }

    /// <summary>
    /// Update is called every frame, if the MonoBehaviour is enabled.
    /// </summary>
    void Update()
    {
        int yata;
        if (true)
            yata = MyMethod(yata, yata, yata);
        else
        {
            yata = MyMethod(yata, yata, yata);
            yata = MyMethod(yata, yata, yata);
        }
    }

```

Nomination

- Les objets (classe, attributs, variables, fonctions, macros, types, fichiers ou répertoires) doivent avoir les noms les plus explicites ou mnémoniques.
- Les noms composites ne doivent être séparés par un caractère spécial ou un autre, mais doivent commencer par une majuscule.
- Les classes doivent commencer par une majuscule. Ex : MaClasse.
- Les attributs commencent par un '_'. Ex : _monAttributs.
- Les variables commencent par une minuscule. Ex : maVariable.
- Les méthodes commencent par une majuscule. Ex : MaMethode.

Déclarations / Affectations

- On sautera une ligne entre la déclaration de variable et les instructions. On met une variable par ligne.
- Il est interdit d'affecter et de déclarer une variable en même temps, excepté lorsque l'on manipule des variables statiques ou globales.
- On alignera les déclarations avec des tab.

Structures de contrôle

- Une structure de contrôle sera toujours suivie d'un retour à la ligne.

Paramètres

- S'il y a plus de quatre paramètres passé dans une méthode, il faut les mettre à la ligne.

Espaces

- Un espace derrière la virgule.
- Pas d'espace entre le nom d'une fonction et la '('.
- Un espace entre un mot clé et la '('.
- Pas d'espace après un opérateur unaire.
- Tous les opérateurs binaires et ternaires sont séparés des arguments par un espace de part et d'autre.

Headers

- Le header doit contenir le nom du fichier.
- Une description du fichier et un contexte d'utilisation.
- La version.
- Le nom de la personne qui a créé le fichier.
- La date de création du fichier.
- La date de la dernière modification.
- Le nom de la personne qui a modifié le fichier en dernier.

```
/*
*   File : exemple.cs
*   Description : this file is a test about code standard.
*   Version : 1.0.1
*   Created by : Jonathan Bihet
*   Created Date : 21/02/2013
*   Modification Date : 21/02/2013
*   Modified by : Jonathan Bihet
*/

using UnityEngine;
using System.Collections;
```