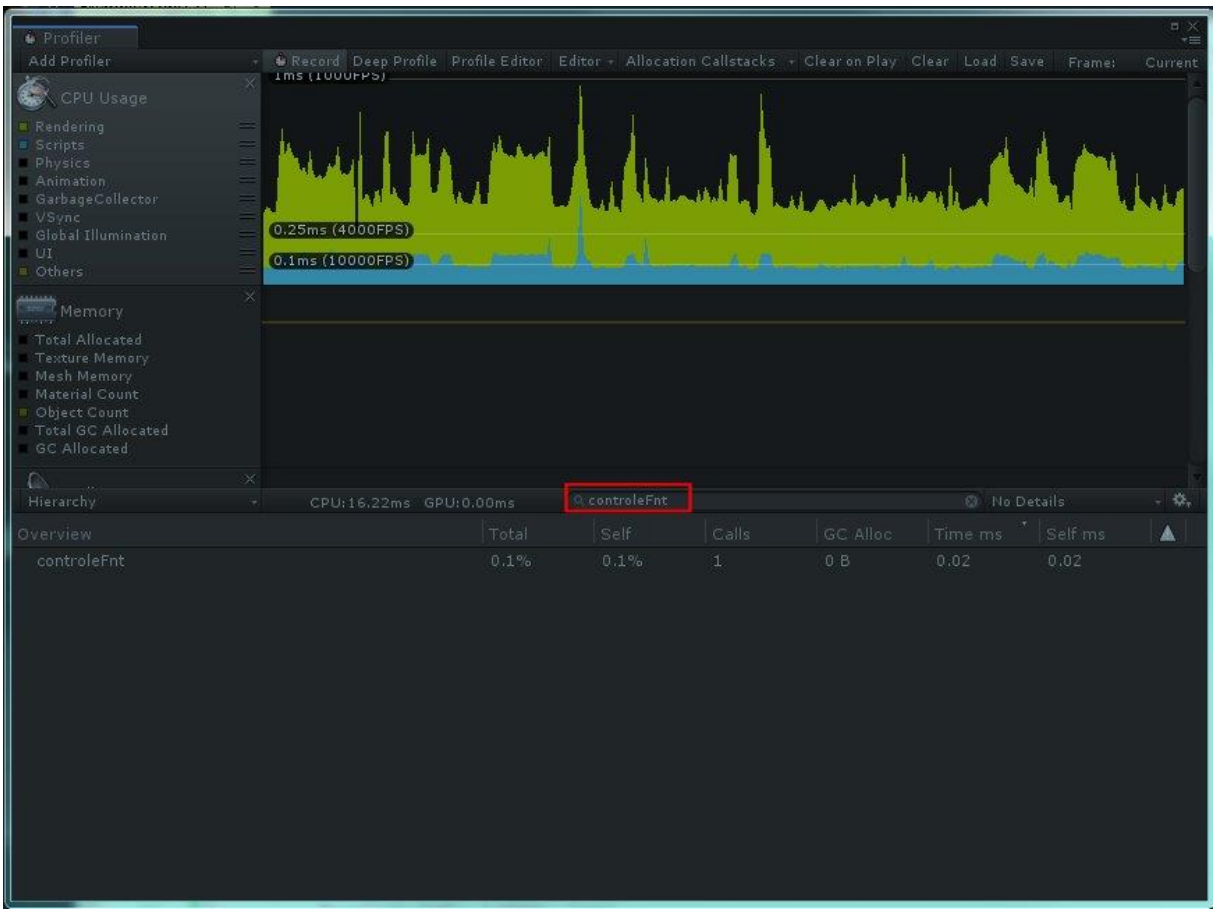
Tip #1: cibler les performances d'une fonction

SCRIPTING

Cibler et visualiser dans le *Profiler* les performances d'une fonction/méthode:

```
using UnityEngine;
using UnityEngine.Profiling;

public class ExempleScript : MonoBehaviour
{

    // Update is called once per frame
    void Update()
    {
        Profiler.BeginSample("controleFnt");

        FonctionCalcul();

        Profiler.EndSample();
    }

    void FonctionCalcul()
    {
        float value = 0f;
        for(int i=0; i<200; i++)
        {
            value += Mathf.Pow(Mathf.Sqrt((float)i), 10.0f);
        }
    }
```

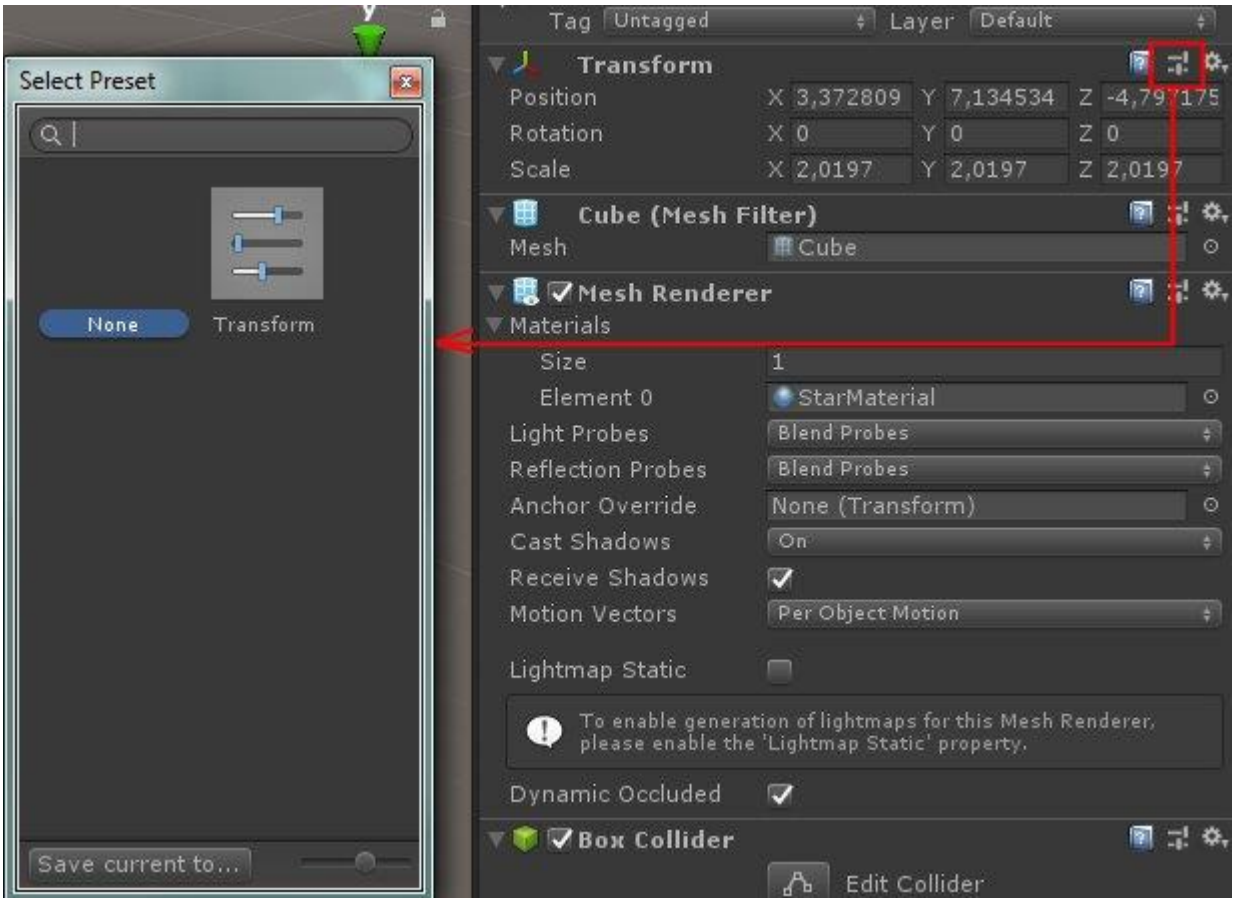
Résultat dans le *Profiler*:



Tip #2: prédéfinir les valeurs des components

EDITOR

Il est possible de prédéfinir les valeurs des components (y compris pour les scripts).
Par exemple, pour appliquer les mêmes valeurs à certains *transforms*, sélectionnez le GameObject de référence, puis allez sur le component souhaité (dans l'exemple un Transform), cliquez sur le symbole de réglage en haut à droite du component. Une fenêtre "Select Preset" s'affiche :

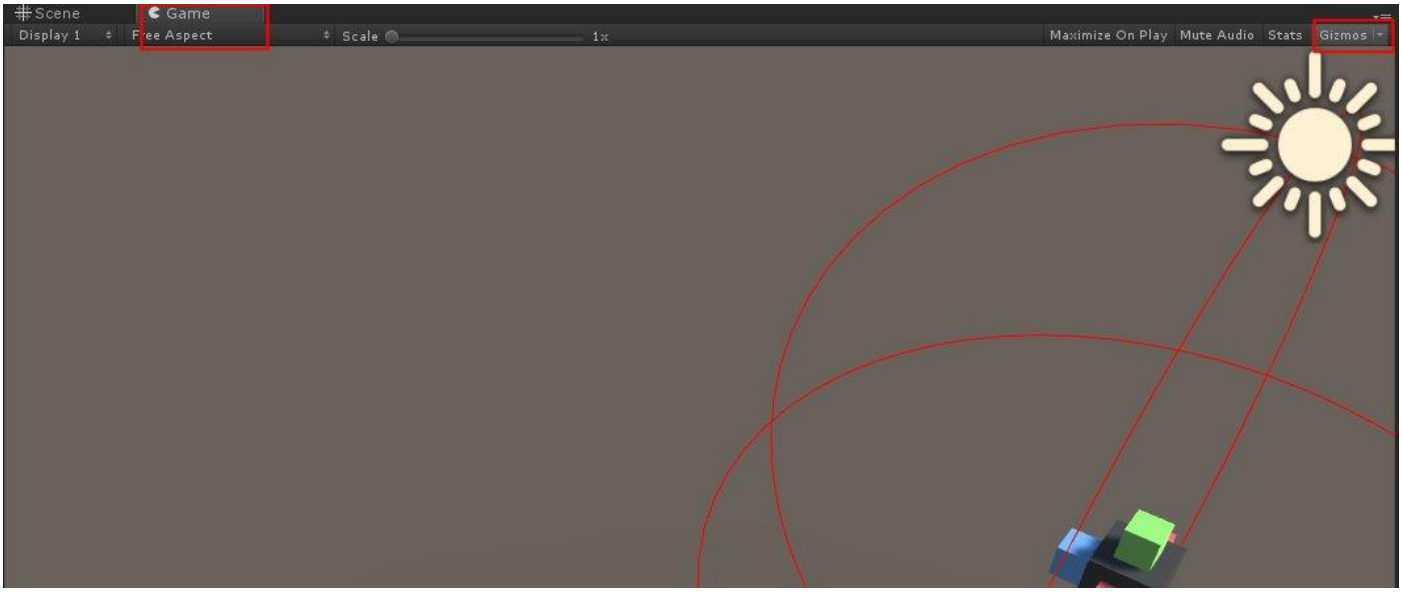


Effectuez « Save Current to… » pour sauvegarder le réglage du component (entrez le nom de votre choix).
Puis, sur les GameObjects où l'on souhaite reproduire les mêmes réglages, dans l'inspector au niveau du même type de component, allez sur l'icône de réglage, puis dans la fenêtre de sélection ouverte, choisir le réglage prédéfinis à appliquer:

Tip #3: afficher les gizmos dans la fenêtre Game

EDITOR

Pour afficher vos *Gizmos* en mode Play dans la fenêtre *Game* , activez en haut à droite le bouton "Gizmos":



Tip #4: limiter la fréquence d'éxecution de certaines parties du code

SCRIPTING

Dans certaines situations, au sein des fonctions *Update*, il peut parfois être intéressant de limiter la fréquence d'exécution de certaines parties du code, relativement couteuses en temps.

Par exemple, pour une exécution une fois toute les 3 frames:

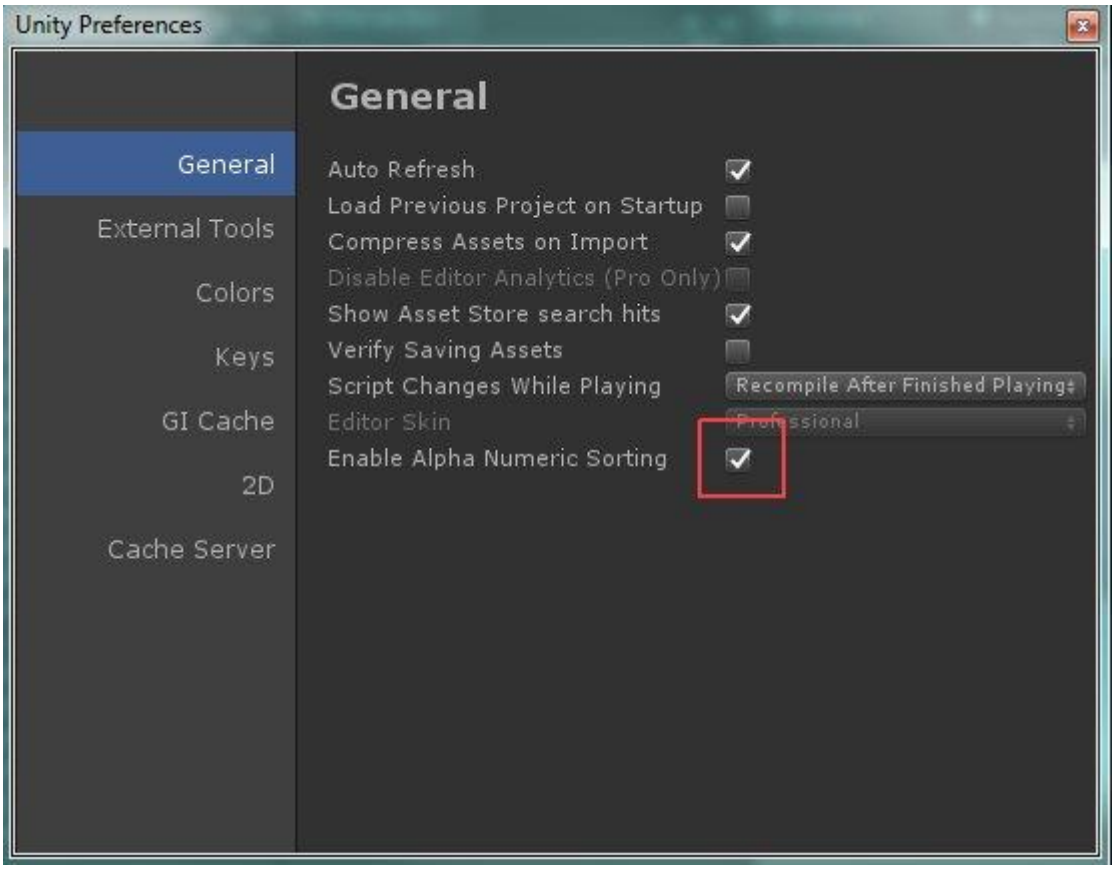
```
private int interval = 3;

private void Update()
{
    if(Time.frameCount % interval == 0)
    {
        // vos codes
    }
}
```

Tip #5: organisation par ordre alphabétique dans la fenêtre "Hiérarchy"

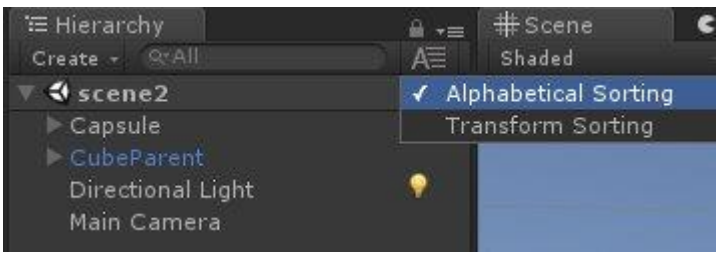
EDITOR

Pour organiser la liste des GameObjects dans la fenêtre "Hierarchy" par ordre alphabétique:

- vérifiez dans les préférences que l'option "*Enable Alpha Numeric Sorting*" est bien cochée.



- en suite, allez dans le menu en haut à droite de la fenêtre 'Hierarchy', puis sélectionnez "*Alphabetical Sorting*"

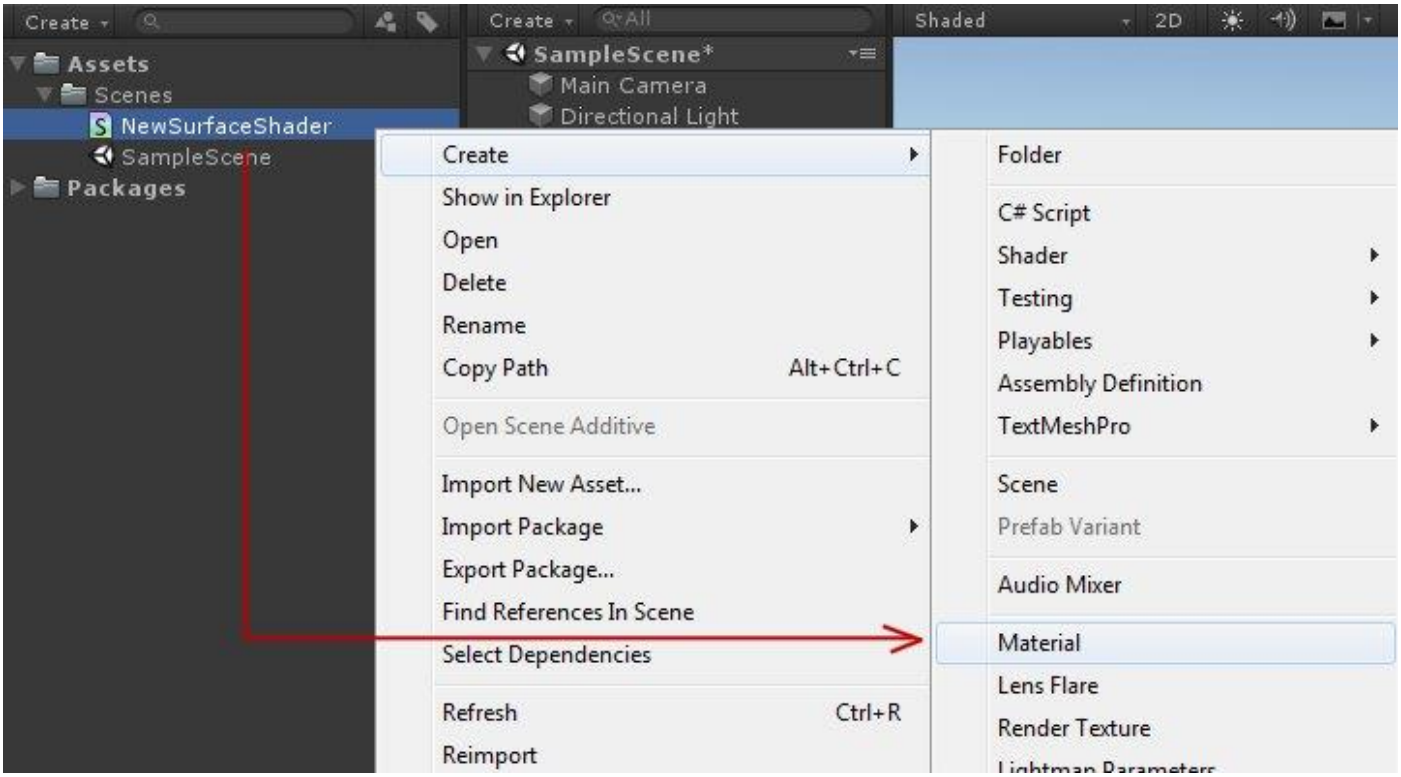


Tip #6: Créer un Material directement avec un Shader spécifique.

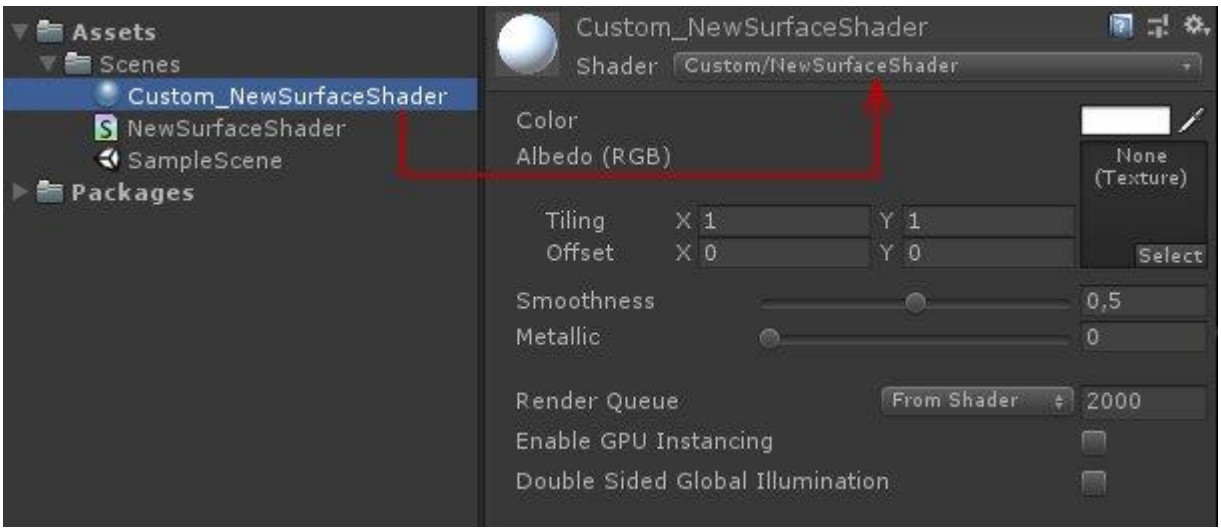
EDITOR

Sélectionnez le *Shader* souhaité :

Puis click droit -> menu de création -> puis *Material* :



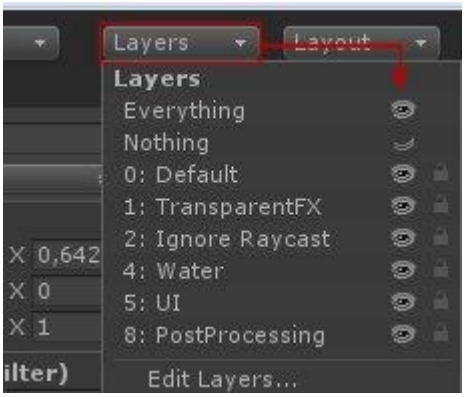
Material alors créé :



Tip #7: Masquer l'affiche de certains GameObject dans la fenêtre scène.

EDITOR

Pour faciliter le travail sur certaines scènes très chargées, il est possible de masquer certains groupes de *GameObjects* en fonction de leur *layer*. De la même façon, il est possible de bloquer la possibilité de sélection (picking) sur certains layers.
Pour cela, allez dans le menu *Layers* (en haut à droite dans l'éditeur), puis choisissez les *Layers* à afficher/masquer (petit symbole en forme d'œil) ou pour limiter le picking (petit symbole en forme de cadenas):



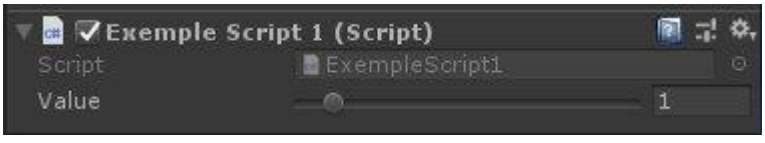
Tip #10: Un slider pour vos données dans l'inspector.

SCRIPTING

Utilisation d'un slider pour borner vos valeurs au sein de l'inspector:

```
    [Range(0f, 10f)]
     public float value = 1f;
```

Dans l'*Inspector* nous aurons :



Tip #8: Aligner automatiquement la camera sur la view scene.

SCRIPTING

Une astuce, sous forme de script, proposée par **ZJP**, et qui permet d'aligner automatiquement le camera sur la vue scène, y compris en mode Play. Pour ce faire, créer par exemple un *Empty*, et placer y le script ci-dessous:

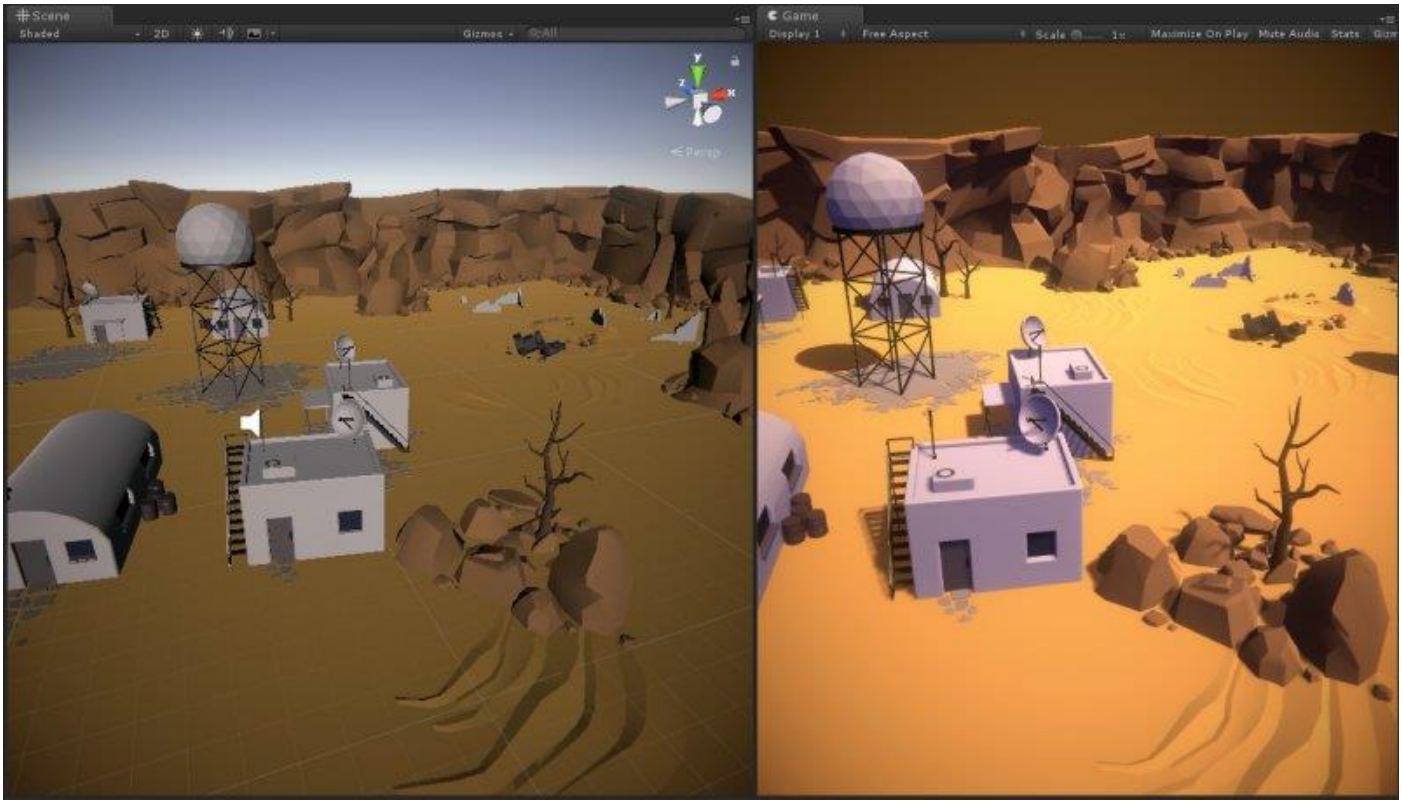
```
using UnityEngine;

public class COPIEcam : MonoBehaviour {

    [SerializeField]
    private Camera mCamera;

#if UNITY_EDITOR

    // Aligne la camera y compris en runtime
    public void OnDrawGizmos()
    {
        if (Event.current.type == EventType.Repaint)
        {
            AlignSceneCamera();
        }
    }

    void AlignSceneCamera()
    {
        var sv = UnityEditor.SceneView.lastActiveSceneView;
        if (sv != null)
        {
            mCamera.transform.position = sv.camera.transform.position;
            mCamera.transform.rotation = sv.camera.transform.rotation;
        }
    }

#endif
}
```

Pour effectuer un alignement uniquement en mode édition (non pris en compte en mode Play), changer la condition

```
if (Event.current.type == EventType.Repaint)
```

Par

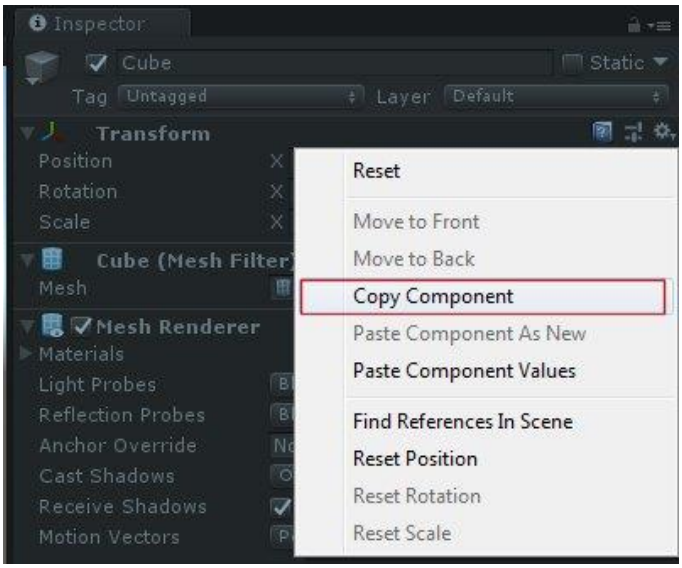
```
if (!Application.isPlaying && Event.current.type == EventType.Repaint)
```



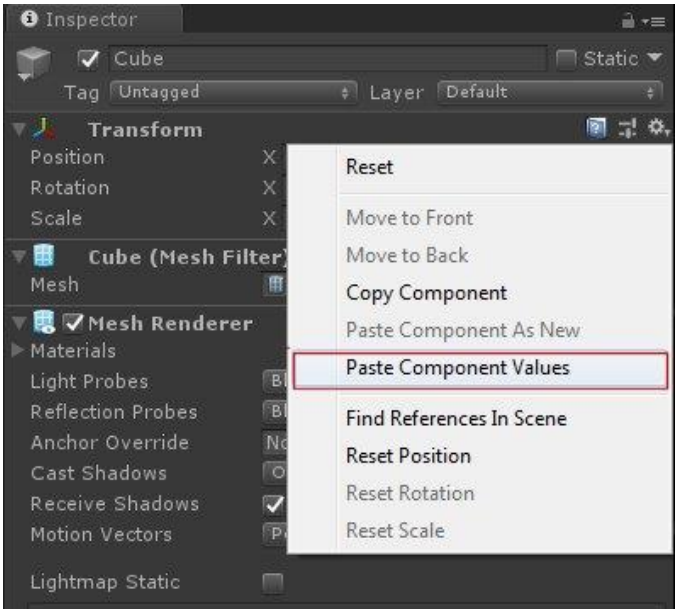
Tip #9: Récupérer les changements opérés en mode Play sur le component d'un GO

EDITOR

En mode Play, au niveau de l'inspector, aller sur le component souhaité du GameObject sélectionné, puis sur la roue dentée (en haut à droite du component) faire *Copy Component*:



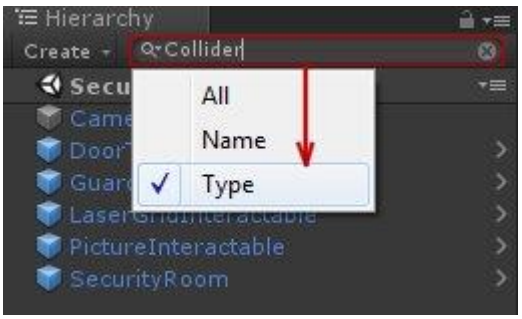
Puis, une fois sorti du mode Play, sur le même type de component (quelque soit le GameObject), cliquez la roue dentée, puis *Paste Component Values*:



Tip #11: Trouver dans la scène tous les GO équipés d'un Component précis.

EDITOR

Dans la fenêtre *Hierarchy*, Il est possible de filtrer les GameObjects par type de Component:



Tip #12: Passage des fenêtres de travail en plein écran.

EDITOR

Pour plus de confort, Il est possible de maximiser n'importe quelle fenêtre de travail au sein de l'éditeur, par un simple appui sur les touches "*Shift + Espace*". Un second appuis ramène la fenêtre à sa position et format initial.

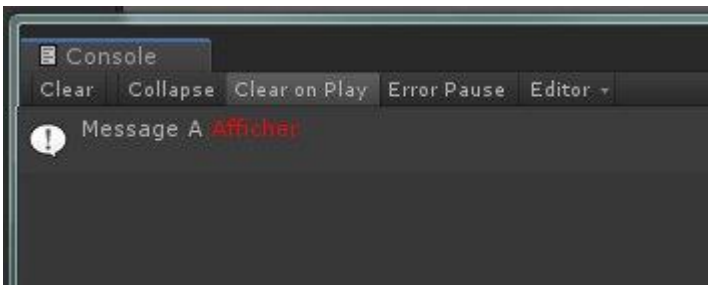
Tip #13: Mettre de la couleur dans ses Logs.

SCRIPTING

Mettez de la couleur dans vos Log. Par exemple :

```
Debug.Log("Message A <color=red>Afficher</color>");
```

Donnera :



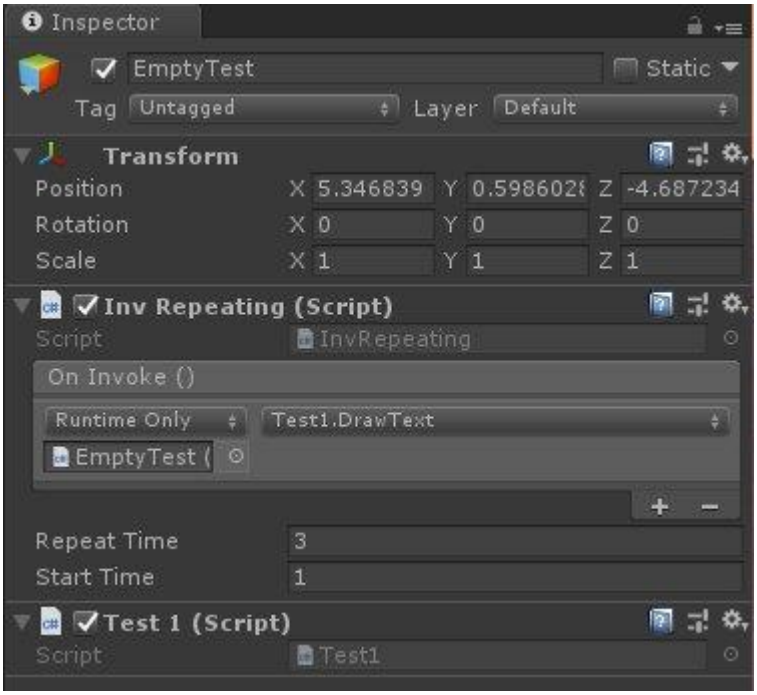
Lien utile: Rich Text.

Tip #14: un système d'événement dans l'inspector

SCRIPTING

Système d'événement dans l'inspector, basé sur InvokeRepeating:

```
using UnityEngine;
using UnityEngine.Events;

public class InvRepeating : MonoBehaviour {

    public UnityEvent onInvoke;
    public float repeatTime = 3f;
    public float startTime = 1f;

    void Start()
    {
        InvokeRepeating("Execute", startTime, repeatTime);
    }

    void Execute()
    {
        onInvoke.Invoke();
    }
}
```



Tip #15: Organiser les gameobjects dans la hierarchy

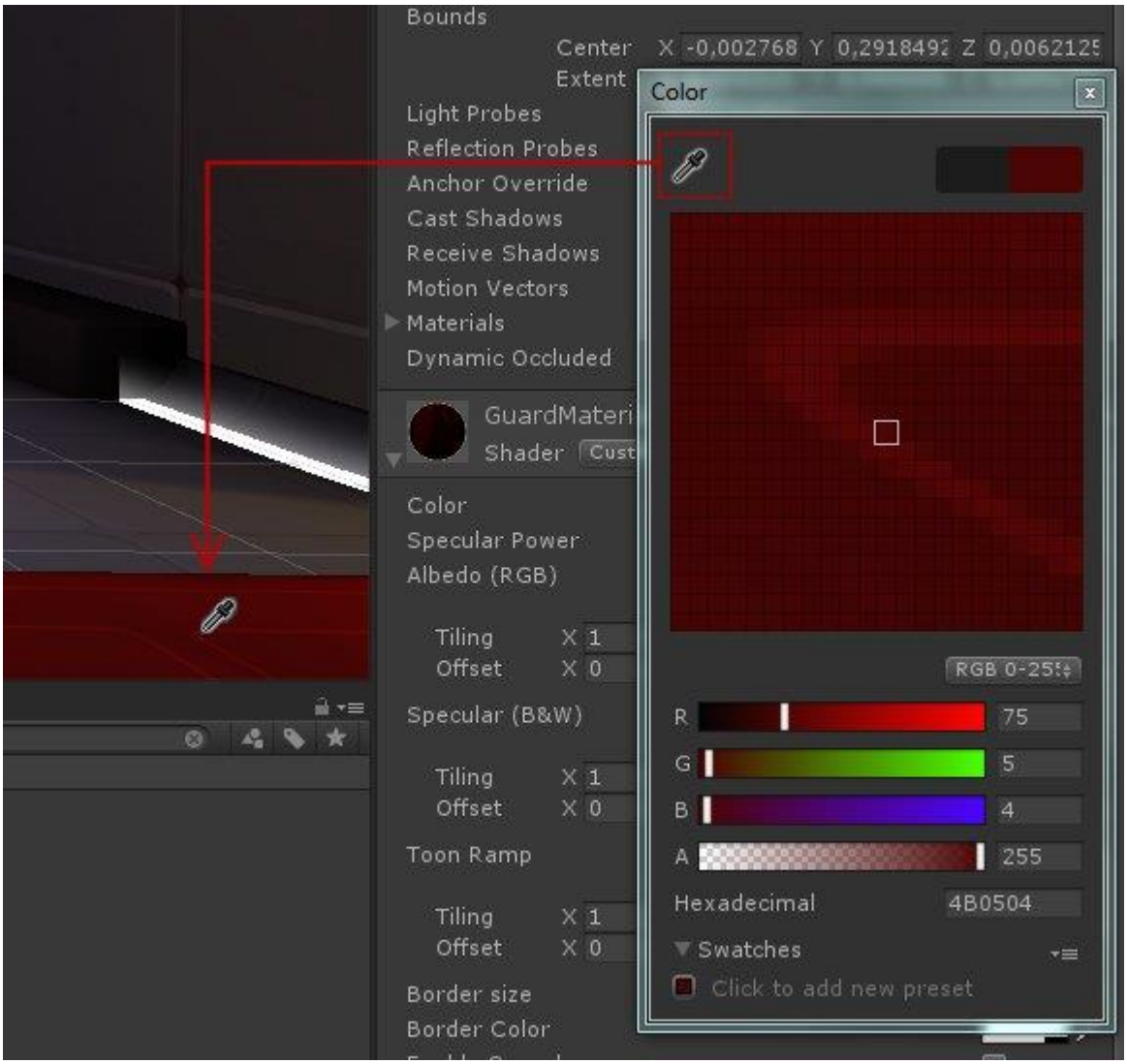
EDITOR

Pour plus de clarté en termes d'organisation dans la liste souvent longue des GameObjects dans la fenêtre Hierarchy, créez des GO Empty nommés de façon explicite pour visualiser chaque catégorie:

Tip #16: capter la couleur d'un material

EDITOR

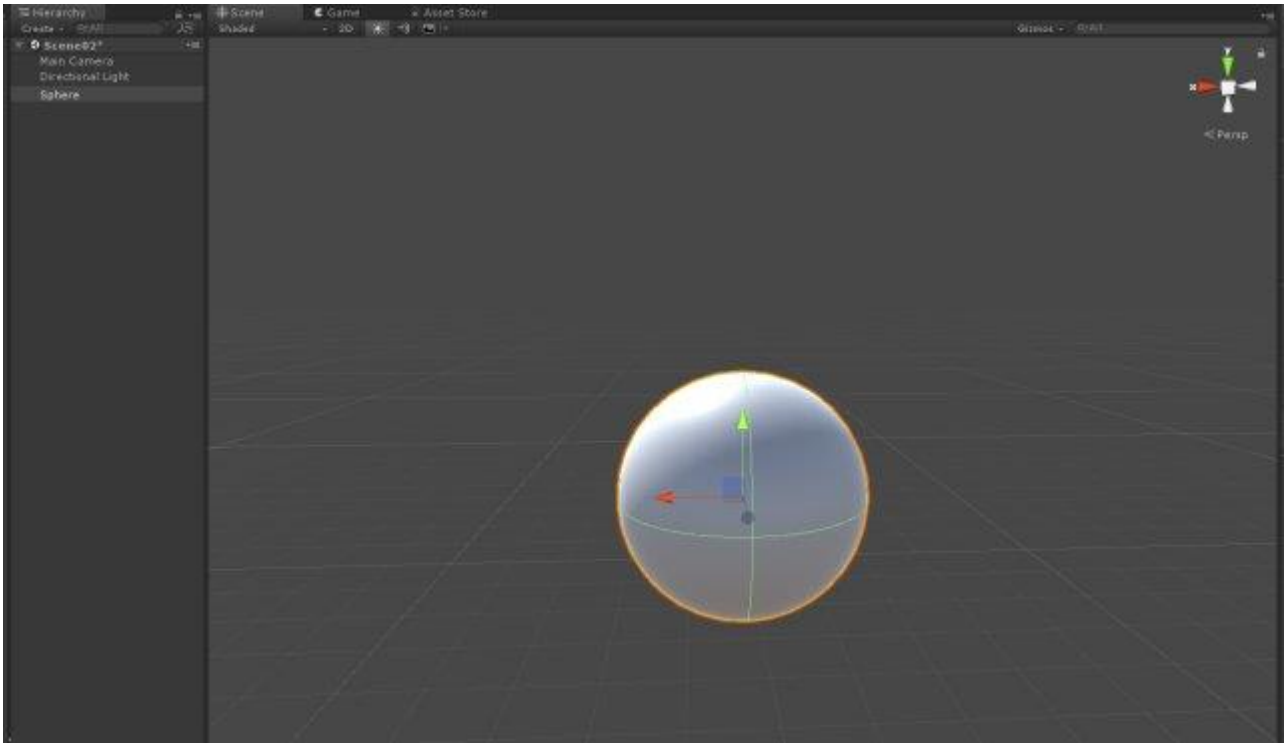
Il est possible de changer la couleur d’un *Material* en allant ‘capter’ directement la couleur souhaitée n'importe où sur l'écran.
Dans l’écran de sélection des couleurs, cliquez sur le petit pinceau en haut à gauche et aller chercher la nouvelle couleur:



Tip #17: centrer la vue scène sur l'objet selectionné.

EDITOR

Pour centrer la vue scène sur un *GameObject*, sélectionnez le GO souhaité dans la partie "*Hierarchy*" ou cliquez sur un GO dans la vue scène, puis le pointeur de la souris se trouvant sur la fenêtre scène, pressez la touche F.



Tip #18: Exécuter un script sans utiliser de GameObject.

SCRIPTING

Il est possible d'initialiser une méthode au moment ou le jeu est chargé, sans action de l'utilisateur. Les méthodes marquées [RuntimeInitializeOnLoadMethod] sont appelées après le chargement du jeu. Juste avant ou après les appels aux méthodes Awake.
Exemple:

```
using UnityEngine;

class MyClass
{
    [RuntimeInitializeOnLoadMethod(RuntimeInitializeLoadType.BeforeSceneLoad)]
    static void MethodLoad1()
    {
        Debug.Log("Avant le lancement du jeu");
    }

    [RuntimeInitializeOnLoadMethod(RuntimeInitializeLoadType.AfterSceneLoad)]
    static void MethodLoad2()
    {
        Debug.Log("Juste après le chargement de la scène et l'exécution des
Awakes.");
    }
}
```

Tip #19: Fusionner deux scènes.

EDITOR

Pour fusionner deux scènes, glisser la scène 1 sur la scène 2:

Puis choisissez l'option souhaitée :

Tip #20: random sur un boolean.

SCRIPTING

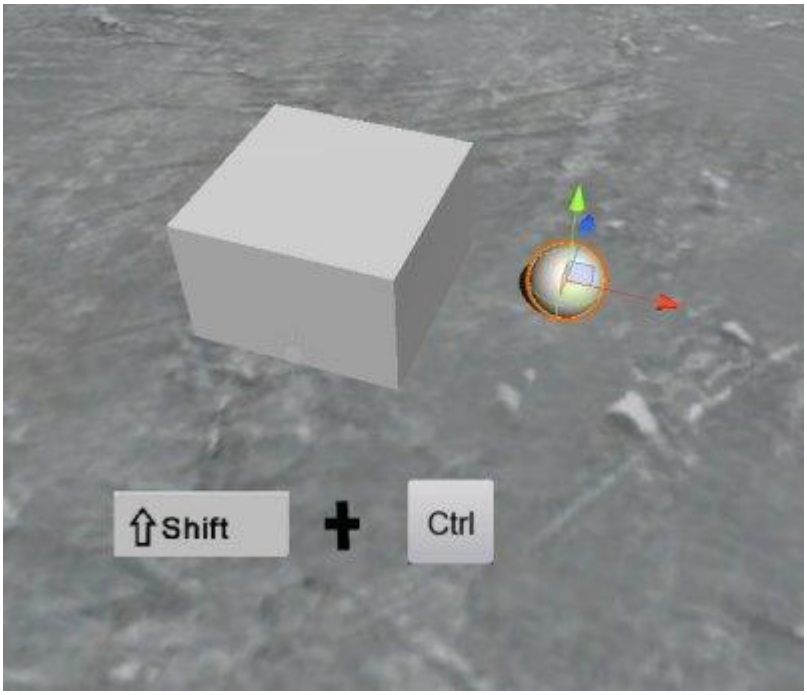
Pour effectuer un random sur un type boolean:

```
        // bool random
        bool typebool = (Random.value > 0.5f);
```

Tip #21: plaquer un GO sur la surface d'un autre

EDITOR

Dans la fenêtre scène, en mode *Move* sur le *GameObject* à déplacer, appuyez simultanément sur Shift et Ctrl. Puis en suite, cliquez et déplacer le GO. Si le GO placer en dessous *possède un collider*, alors il sera plaqué dessus.



Tip #22: masquer ses données dans l'inspector.

SCRIPTING

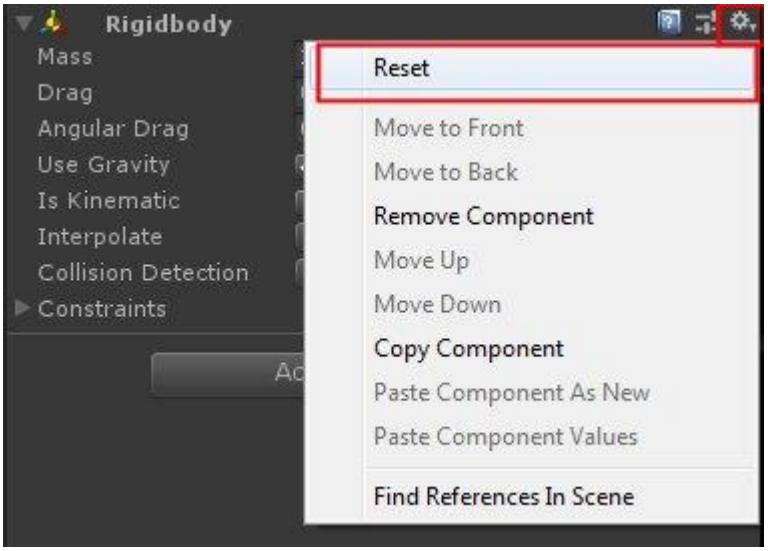
Pour éviter un affichage des données dans l'inspector (y compris en mode Debug):

```
    [HideInInspector]
    public float v;

    [HideInInspector]
    private int i;
```

Tip #23: réinitialiser les paramètres par défaut d'un component.

EDITOR

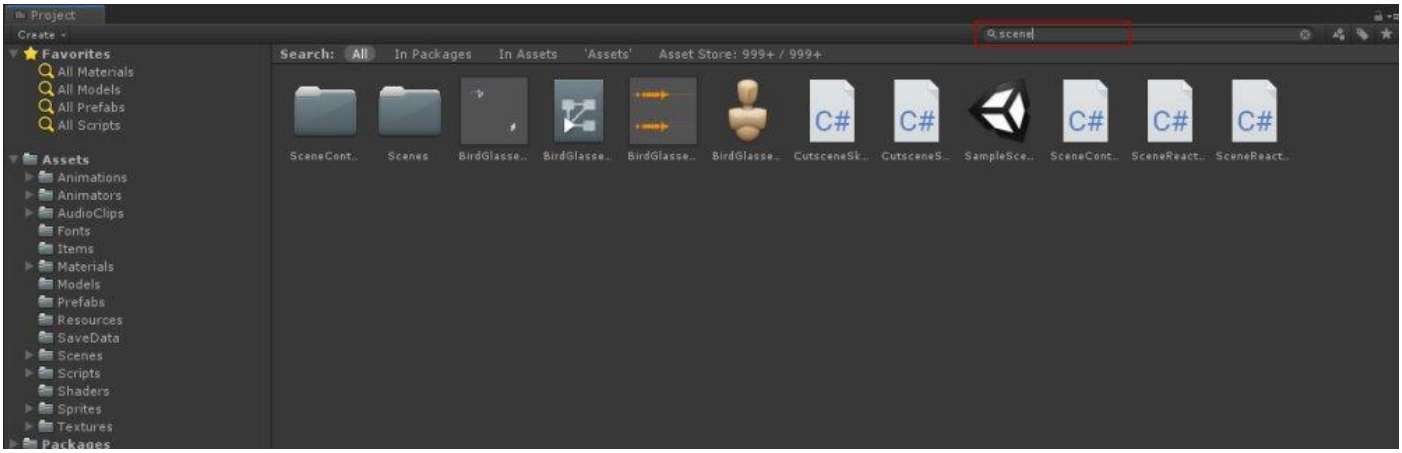
Dans l'inspector, cliquez sur la petite roue dentée située en haut à droite dans la zone du *component* choisi, puis cliquez sur *RESET*:



Tip #24: Filtrer les assets pour faciliter les recherches.

EDITOR

Filtrez les assets, facilitant les recherches dans l'onglet Project:



Tip #25: Visualiser l’évolution d’une donnée sur un graphe.

SCRIPTING

Visualiser l’évolution d’une donnée sur un graphe.

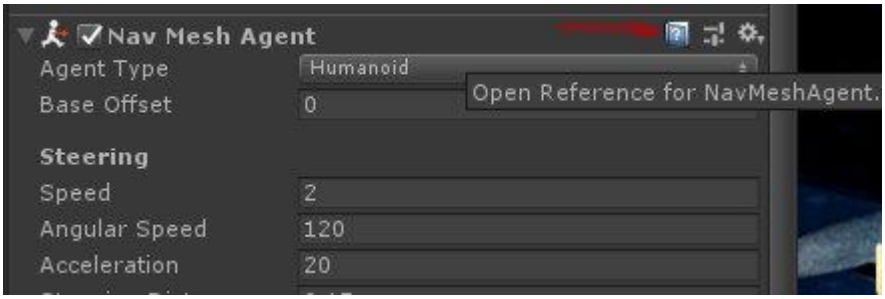
```
public AnimationCurve plot = new AnimationCurve();

// Update is called once per frame
void Update ()
{
        //Ici, donnez à value la donnée à visualiser (ex: fonction sinus)
         float value = Mathf.Sin(Time.time);
         plot.AddKey(Time.realtimeSinceStartup, value);
}
```

Donnera:

Tip #26: Accès direct à la documentation d'un composant.

EDITOR

Dans l'inspector, au niveau du *component*, cliquez sur le petit livre en haut à droite.
Votre navigateur ouvrira la page de la documentation lié à ce *component*.



Tip #27: récupérer les GameObjects d'une scène.

SCRIPTING

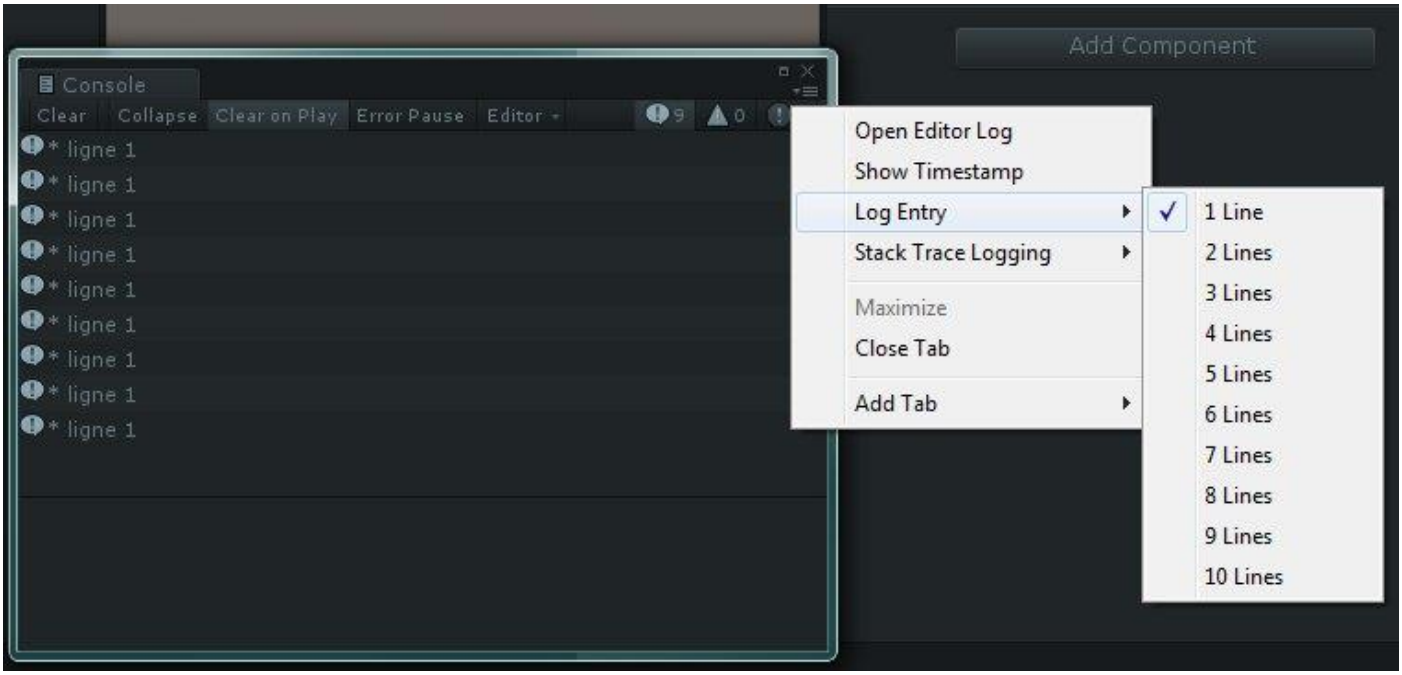
Pour récupérer tous les *GameObjects* actifs (parents/enfants) d'une scène:

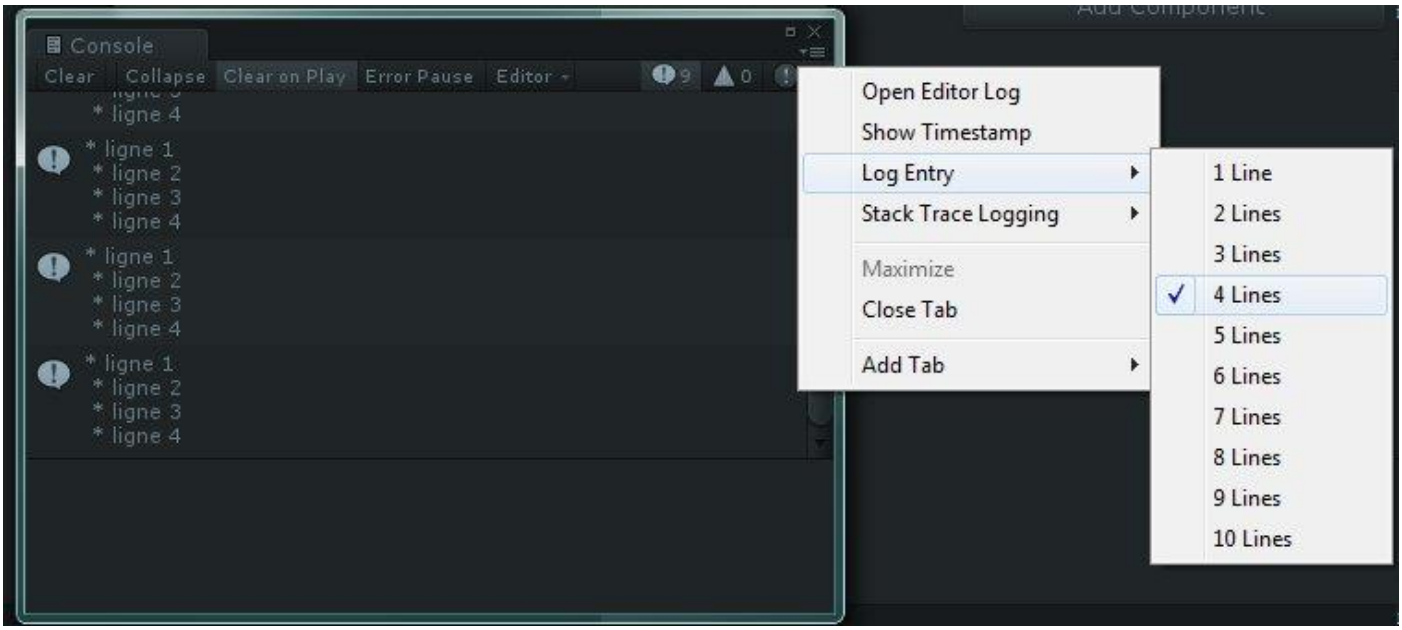
```
GameObject[] gameObjs  = FindObjectsOfType<GameObject>()
as GameObject[];

for (int i=0; i< gameObjs.Length; i++)
{
      Debug.Log(gameObjs[i].name);
}
```

Tip #28: choisir le nombre de lignes à afficher dans la console:

EDITOR

Il est possible de choisir le nombre de lignes à afficher (par message) dans la console:

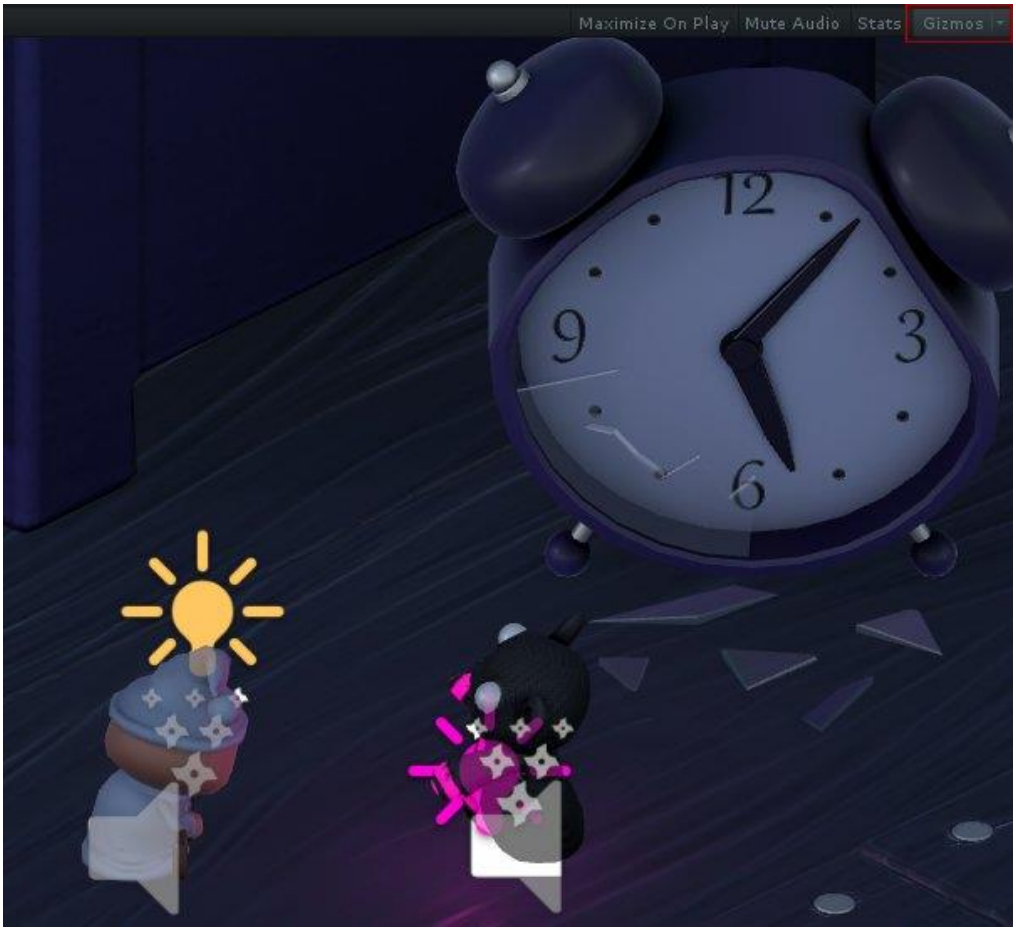




Tip #29: afficher les Gizmos en mode Play.

EDITOR

Pour afficher les *Gizmos* dans la fenêtre Game en mode Play:



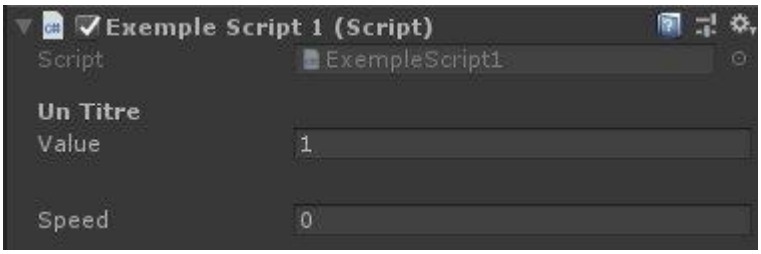
Tip #30: un espace entre vos lignes de données dans l'inspector.

SCRIPTING

Pour insérer un espace entre vos lignes de données dans l'inspector:

```
public class ExempleScript1 : MonoBehaviour {

    public float value = 1f;
    [Space(20)]
    public float Speed;
```

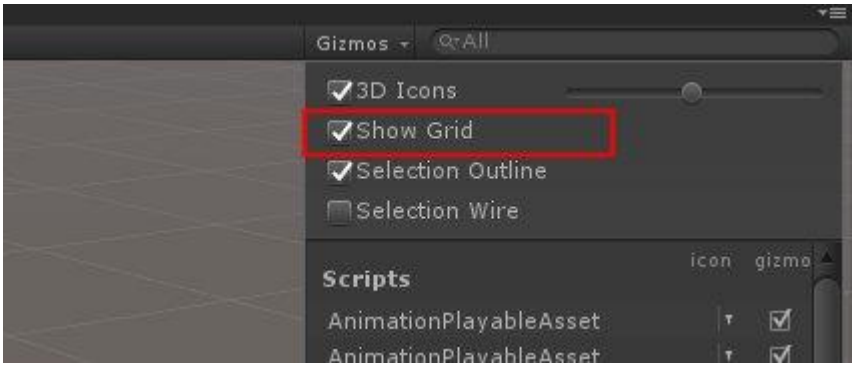
Dans l'inspector, on obtiendra:



Tip #31: masquer ou afficher la grille dans la fenêtre Scene.

EDITOR

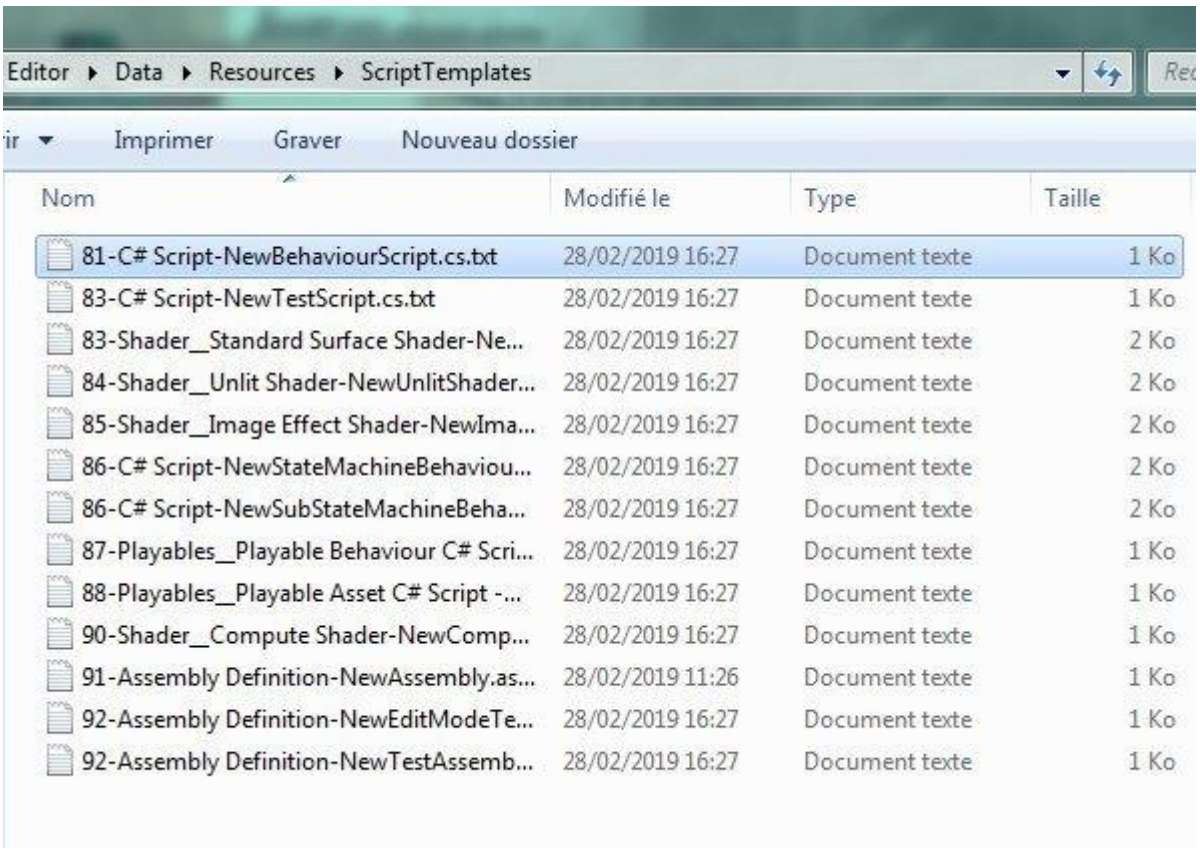
L'option d'affichage de la grille se trouve dans le menu *Gizmo*, situé au niveau de la barre supérieure de l'onglet *Scene*:



Tip #32: modifier le modèle de script par défaut NewBehaviourScript.

SCRIPTING

il est possible de modifier, selon vos besoins, les modèles de scripts d'Unity, en particulier le *NewBehaviourScript.cs*. Les templates sont accessibles dans le répertoire *Editor->Data->Resources->ScriptTemplates*:



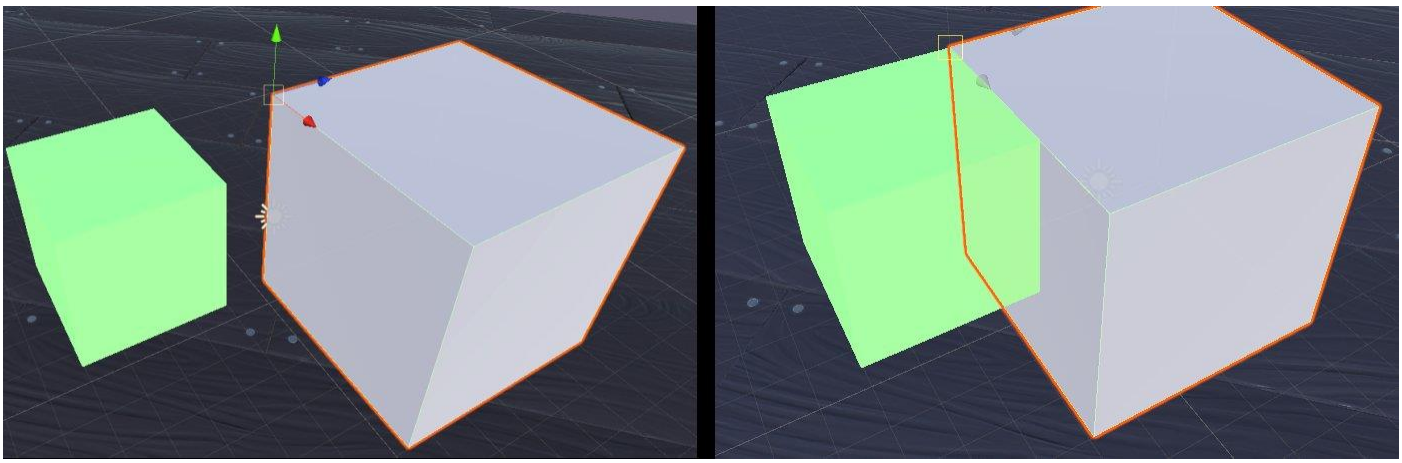
Pour le template *NewBehaviourScript.cs*, le code par défaut est:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class #SCRIPTNAME# : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        #NOTRIM#
    }

    // Update is called once per frame
    void Update()
    {
        #NOTRIM#
    }
}
```

Tip #33: aligner des objets 3D grace à leur sommets (Vertex Snapping).

EDITOR

Pour aligner deux objets 3D grâce à leurs vertices, passez en mode translation, appuyez sur V pour sélectionner le point de l'objet à snapper, puis déplacer l'objet avec la souris vers l'objet cible.



Tip #34: exécuter une partie du code que lorsque le GameObject est visible.

SCRIPTING

Dans certains cas, il est peut être intéressant d'exécuter certaines parties du code uniquement lorsque le GameObject se trouve situé dans le champ de la camera active.

```
if (theRenderer.isVisible)
{
...
}
```

Tip #35: déplier/replier l'arborescence d'un GameObject.

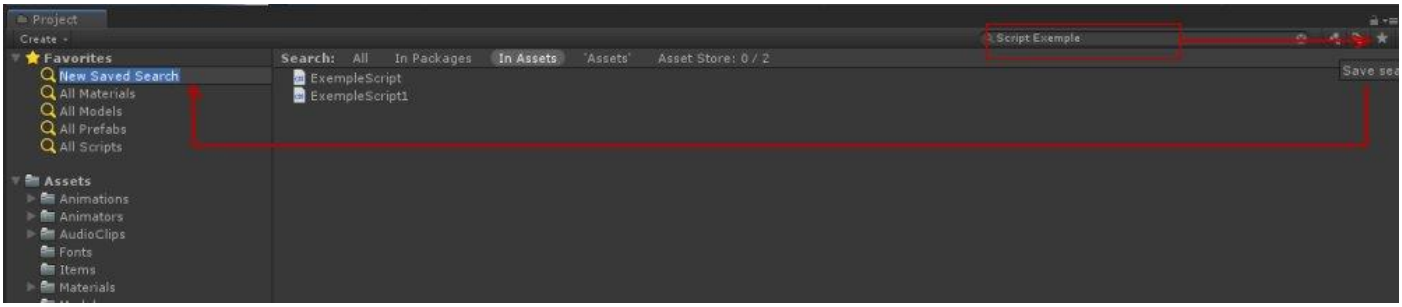
EDITOR

Pour déplier/replier totalement l'arborescence d'un GameObject dans la fenêtre Hierarchy, sélectionner dans la liste le GO et cliquer sur la flèche tout en maintenant la touche ALT.

Tip #36: automatiser certaines recherches dans la zone Project.

EDITOR

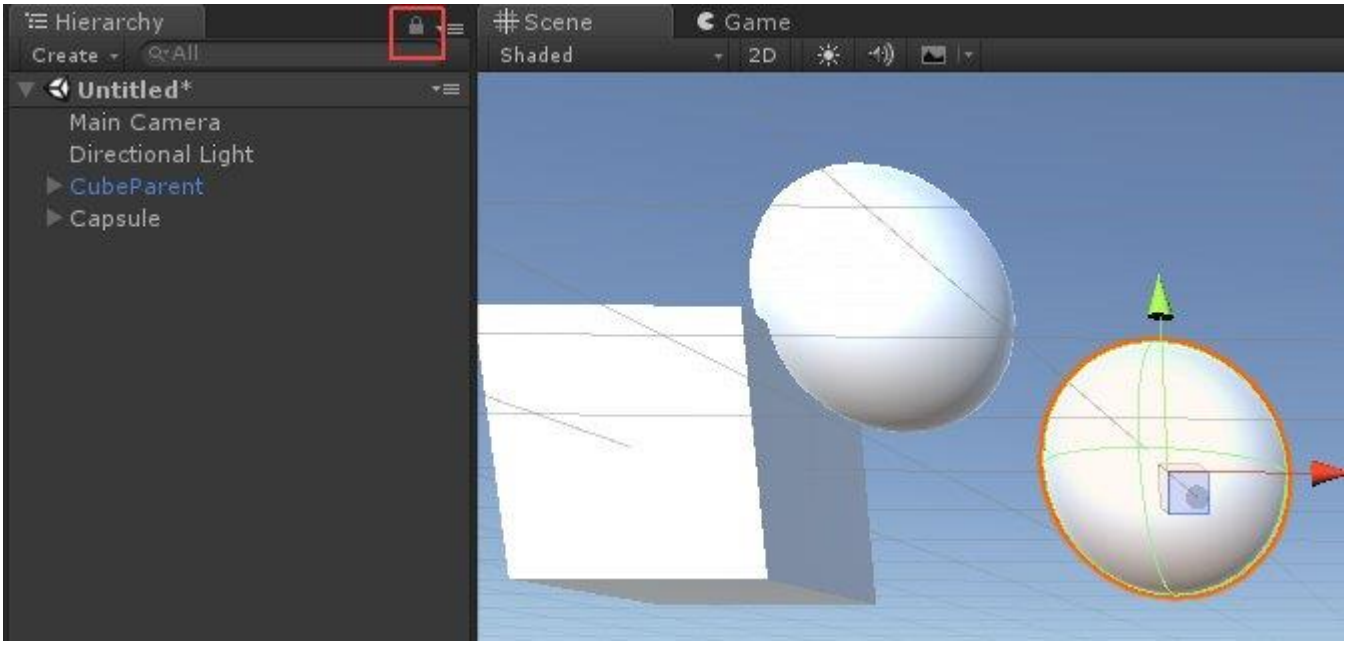
Au sein de projet volumineux et comportant de très nombreux assets, il est possible d'automatiser les recherches courante par la définition de favoris :



Tip #37: selectionner un GO sans dérouler la hiérarchie complète du parent.

EDITOR

Pour sélectionner un GameObject dans la fenêtre scène sans dérouler la hiérarchie complète du parent, verrouillez la fenêtre *Hierarchy* (petit cadenas en haut à droite), puis dans la fenêtre '*scene*' sélectionnez GO souhaité.



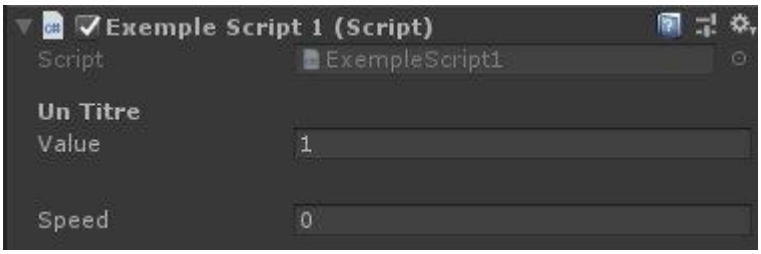
Tip #38: un titre pour vos données dans l'inspector.

SCRIPTING

Pour donner un titre à vos données dans l'inspector:

```
public class ExempleScript1 : MonoBehaviour
{
    [Header("Un Titre")]
    public float value = 1f;
    [Space(20)]
    public float speed;
```

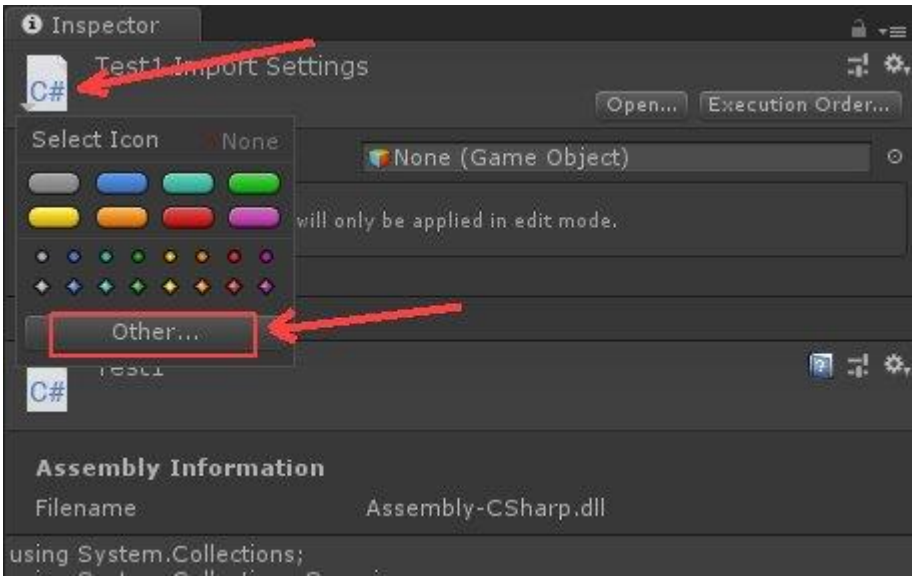
donnera au niveau de l'inspector:



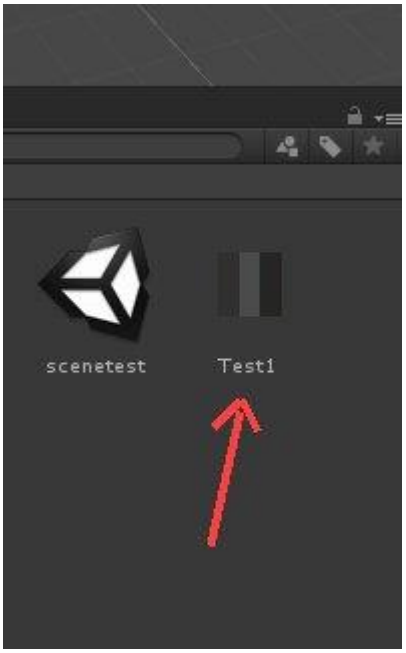
Tip #39: personnaliser les icons de vos scripts.

EDITOR

Pour personnaliser les icônes de vos scripts, sélectionnez le script dans la partie *Project*, dans l'inspector faites "*Select Icon*" en haut à gauche au niveau de l'icône courant, puis sélectionnez "*Other*" et choisissez une image de votre choix.



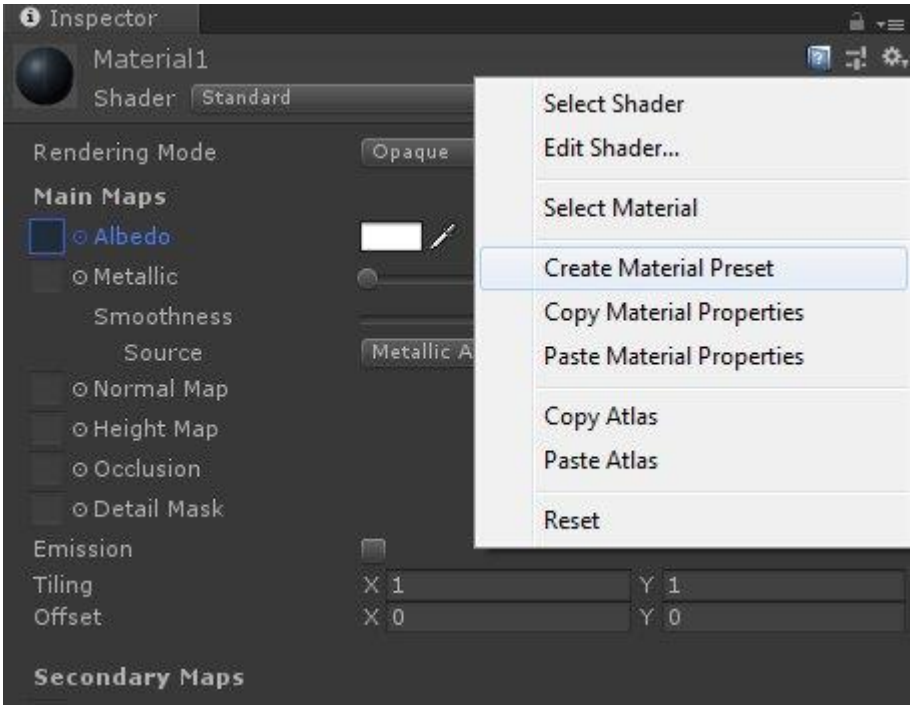
Dans la partie *Project*, votre script sera visualisé avec la nouvelle image:



Tip #40: Creation d'un material à partir d'un material déjà exisant.

EDITOR

Pour créer un *Material* identique à un *Material* déjà existant:



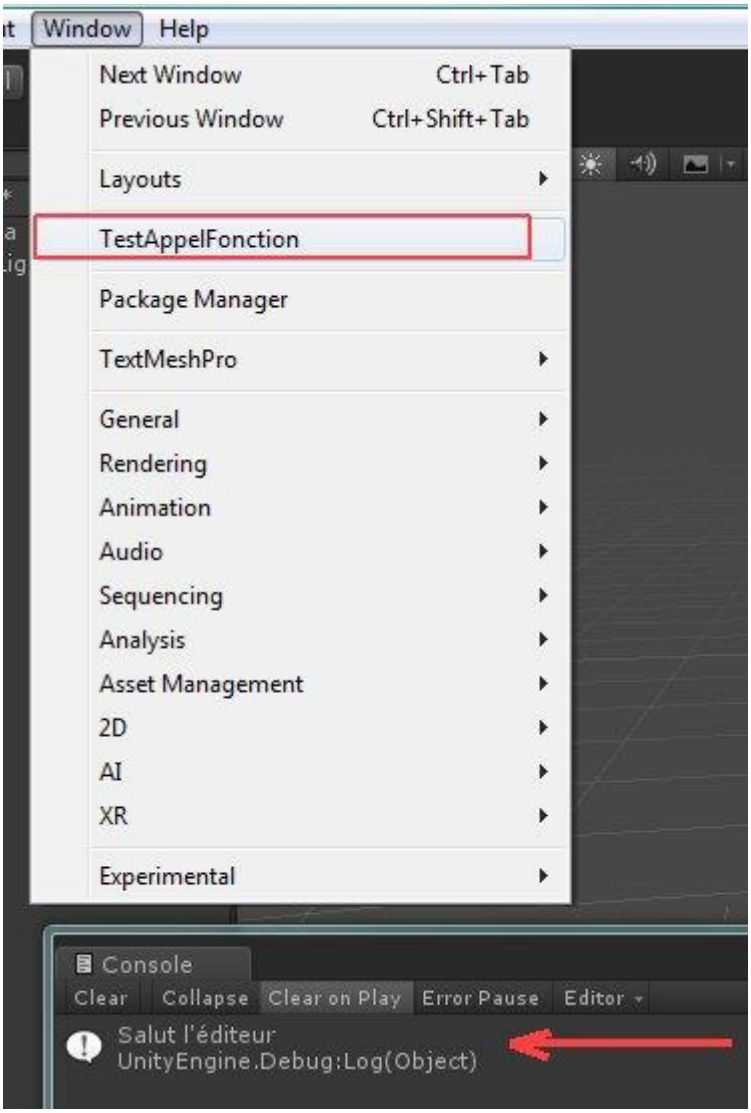
Tip #41: appeler une fonction directement à partir du menu de l’éditeur.

SCRIPTING

Pour appeler une fonction directement à partir du menu de l’éditeur:

```
using UnityEngine;
using UnityEditor;

public class Test1 : MonoBehaviour
{
    [MenuItem("Window/TestAppelFonction")]
    static void FonctionAAppeller()
    {
        Debug.Log("Salut l'éditeur");
    }
```

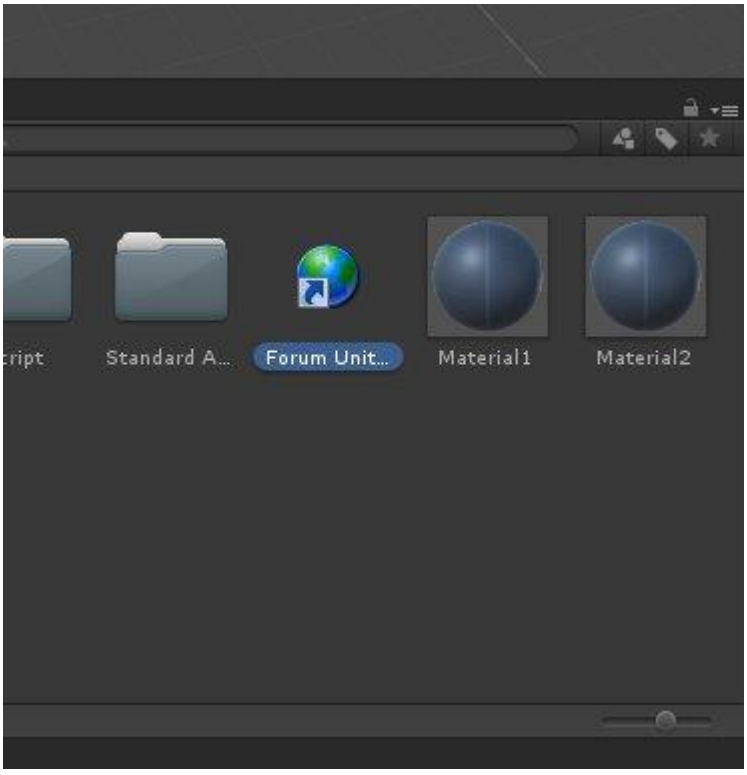
Donnera:



[Tip #42: avoir des liens web directement dans votre projet.]

EDITOR

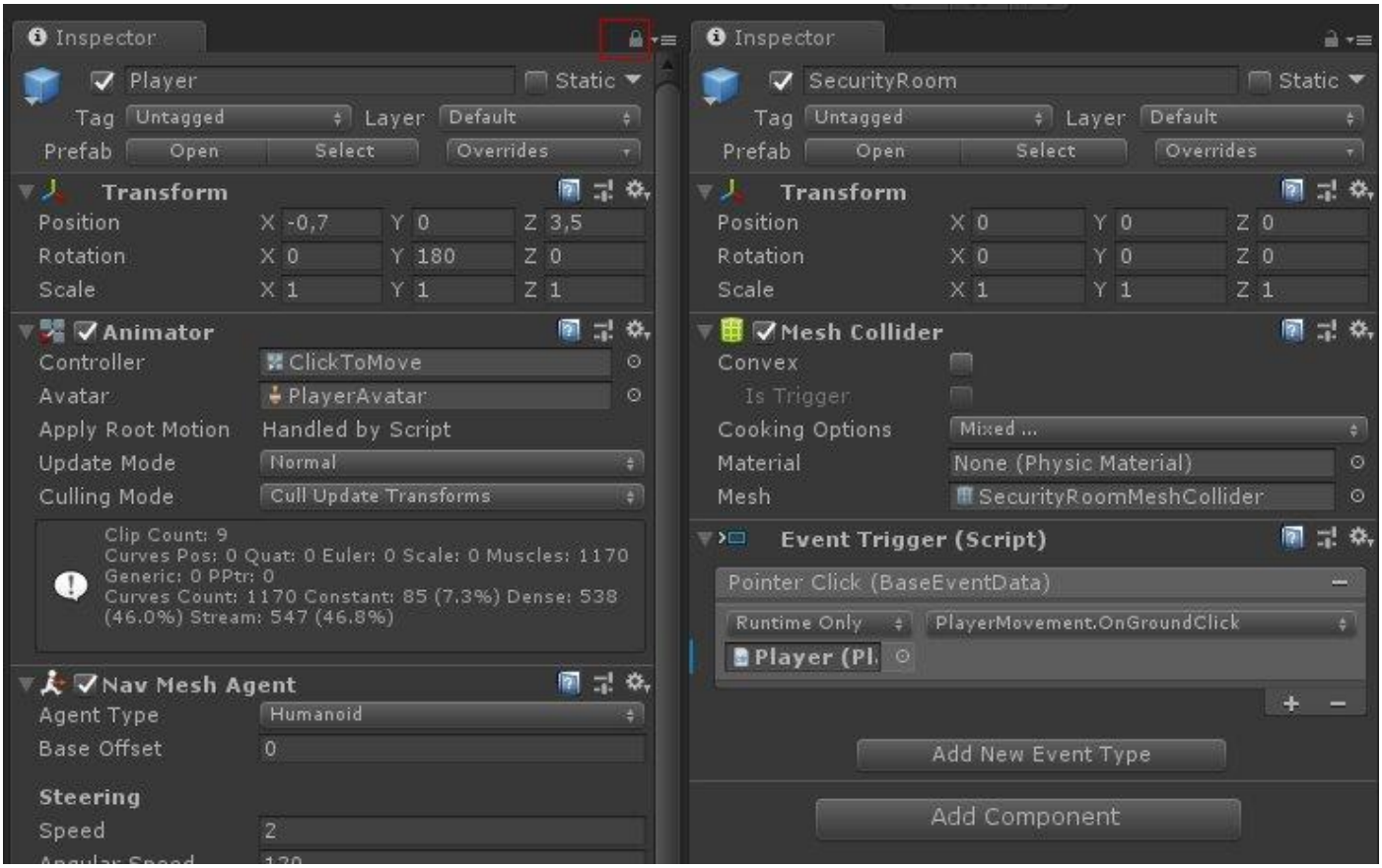
Il est parfois utile d'avoir directement sous la main dans un projet Unity quelques liens utiles.
Il suffit de sélectionnez le lien souhaité, et de faire un drag and drop vers le dossier *Project*:
(un double click sur ce lien ouvrira alors la page dans votre navigateur)



Tip #43: ouvrir un second inspector pour une visu sur deux GameObjects:

EDITOR

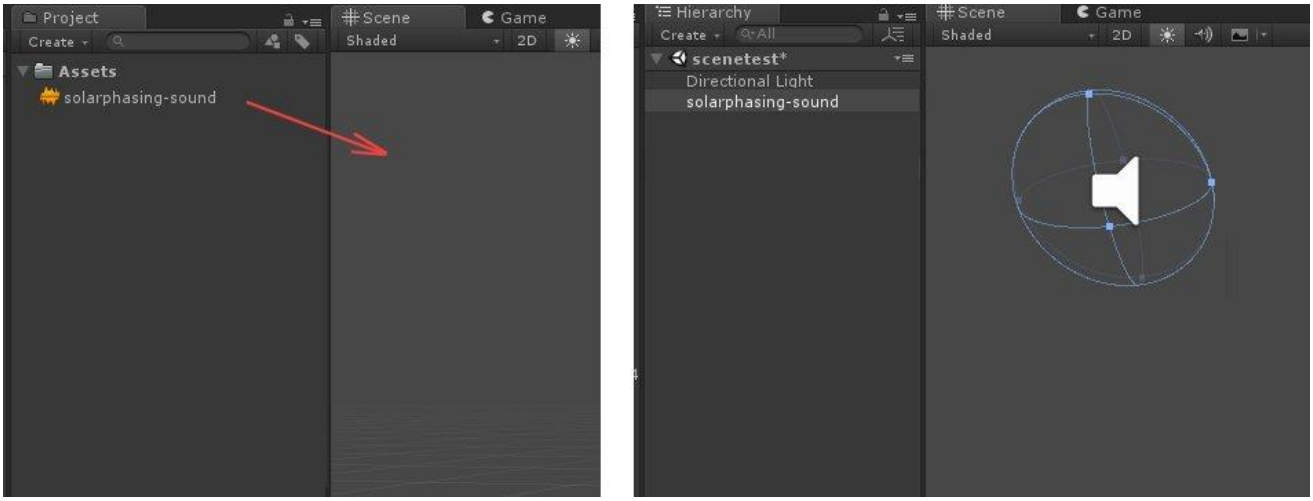
Pour pouvoir consulter simultanément les informations de deux GO, dans deux inspector séparés, verrouillez le premier inspector (petit cadenas en haut à droite), puis ouvrez un second inspector pour le second GO :



Tip #44: créer un GO dans la scene à partir d'un simple audioclip.

EDITOR

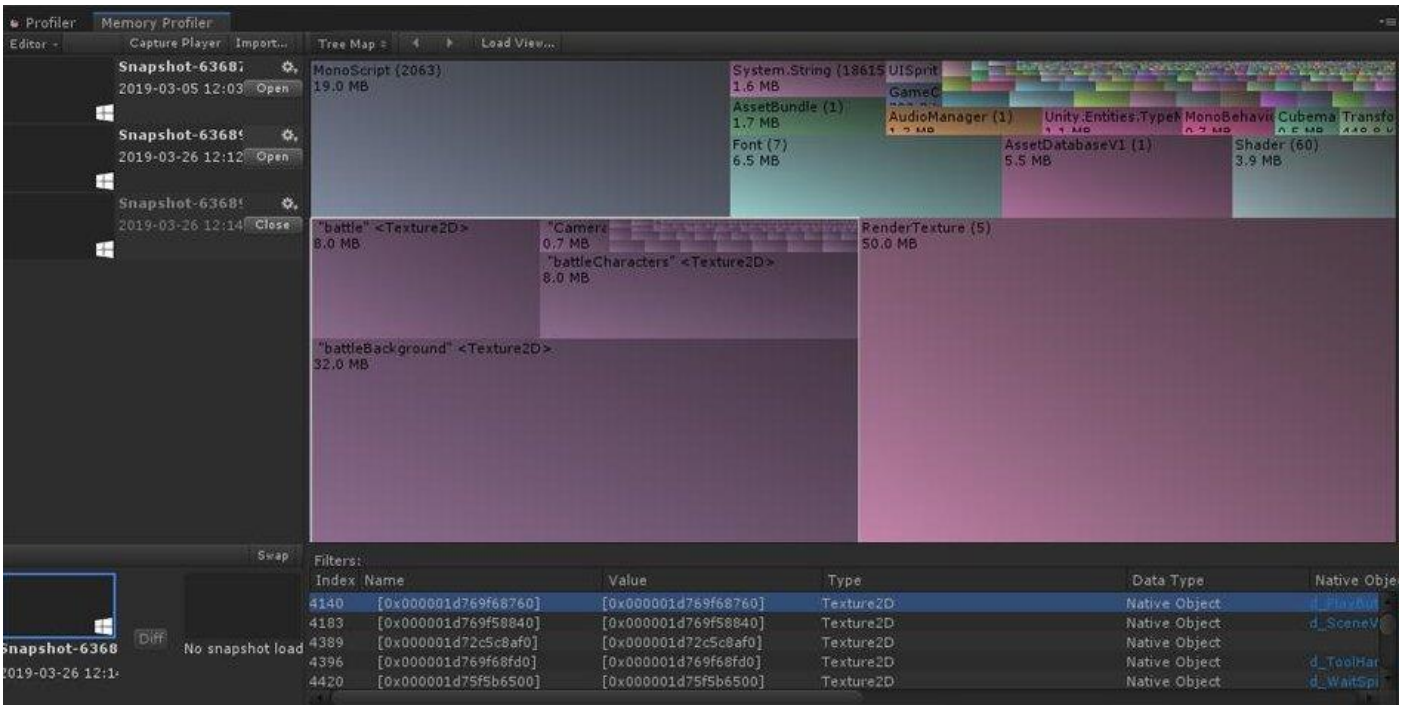
Vous pouvez drag and drop directement dans la scène un audio clip.
Un GameObject "empty" avec un componant AudioSource sera alors automatiquement créé:



Tip #45: osculter la mémoire.

EDITOR

En complément des informations fournies par le Profiler, il est possible d'aller plus loin dans la visualisation de l'utilisation des ressources mémoires. Pour cela, il suffit d'installer le package *Memory Profiler*.



Lien utile: About Memory Profiler

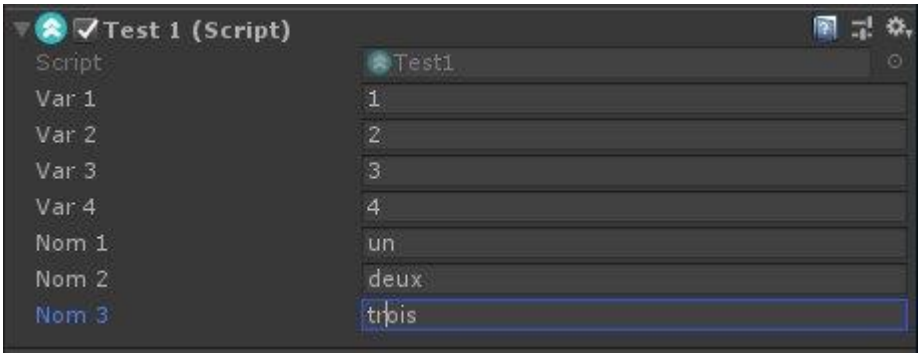
Tip #46: regrouper ses données en catégories dans l'inspector:

EDITOR Addon

Il est possible de regrouper les données par groupe au sein de l'inspector.
Exemple:

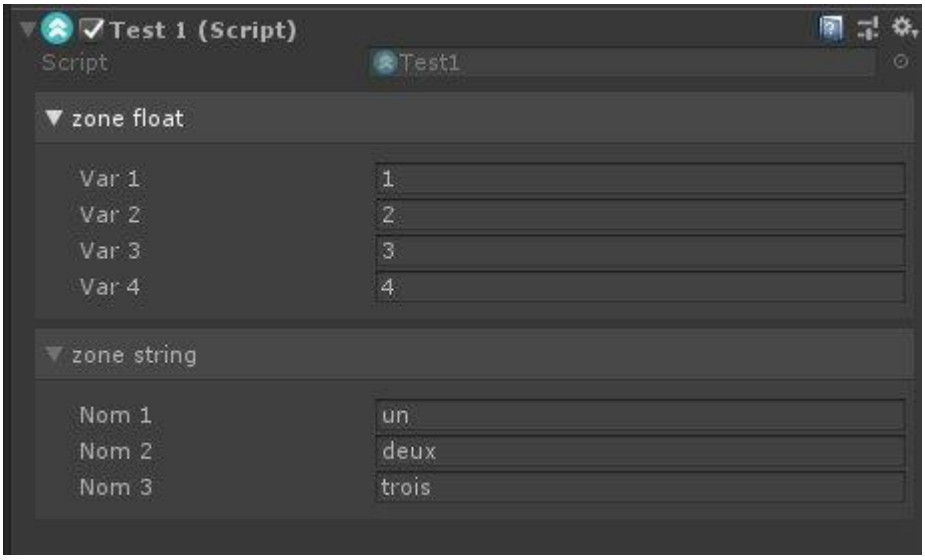
```
public class Test1 : MonoBehaviour{

    public float var1, var2, var3, var4;
    public string nom1, nom2, nom3;
```

on obtient:




```
using UnityEngine;
using Homebrew;

public class Test1 : MonoBehaviour{

    [Foldout("zone float")]
    public float var1, var2, var3, var4;
    [Foldout("zone string")]
    public string nom1, nom2, nom3;
```

on obtient:

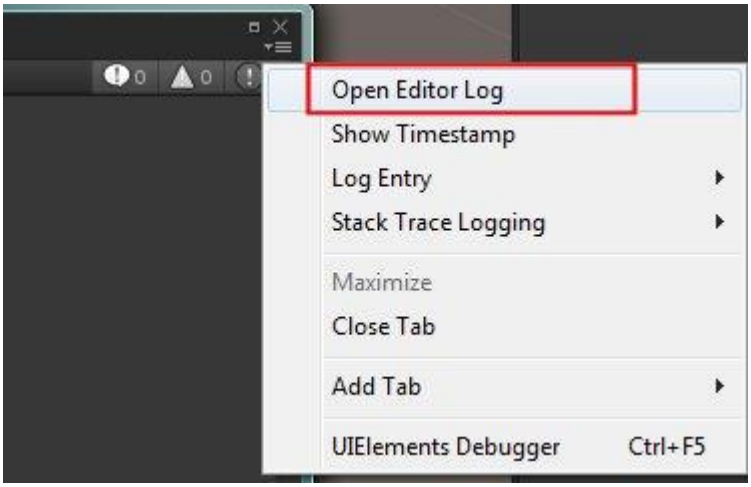


Lien nécessaire: InspectorFoldoutGroup

Tip #47: avoir accès au fichier Editor.log directement depuis la console.

EDITOR

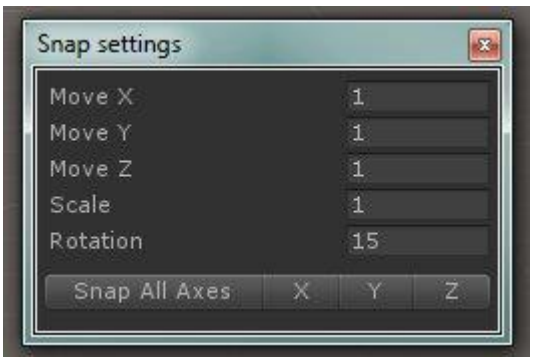
Il est possible de consulter le fichier Editor.log complet depuis l'éditeur. Pour cela, la console étant affichée, dans le menu déroulant proposé en haut à droite sélectionnez '*Open Editor Log*':



Tip #48: déplacer un GameObject en suivant la résolution de la grille.

EDITOR

Les trois actions de base (translation, rotation et changement d'échelle) du tools peuvent se faire avec un pas définis. Maintenez la touche Ctrl enfoncée lors des déplacements de l'objet.
Vous pouvez changer les paramètres de résolution de la grille virtuelle en passant par le menu de l'éditeur *Edit->Snap Setting* :



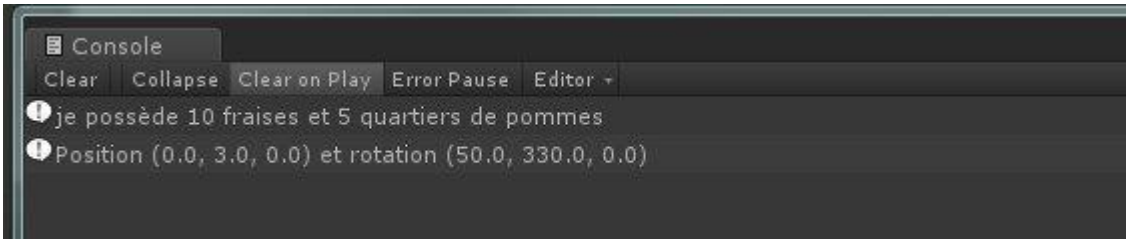
Tip #49: formater facilement vos Log.

SCRIPTING

Debug.LogFormat permet de plus facilement insérer dans vos Logs les données à afficher au milieu de textes descriptifs. Exemple:

```
    int pommes = 5;
    // Use this for initialization
    void Start ()
    {
        Debug.LogFormat("je possède {0} fraises et {1} quartiers de
pommes", 10, pommes);
        Debug.LogFormat("Position {0} et rotation {1} ",
transform.position, transform.rotation.eulerAngles);
    }
```

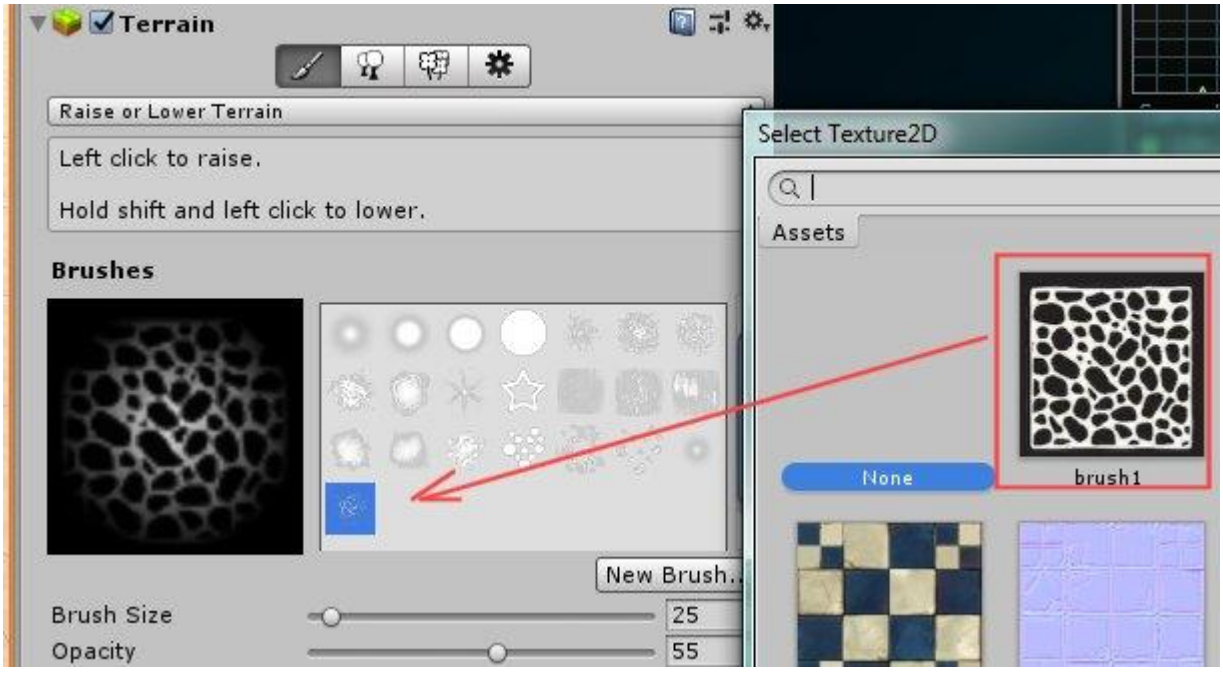
donnera:



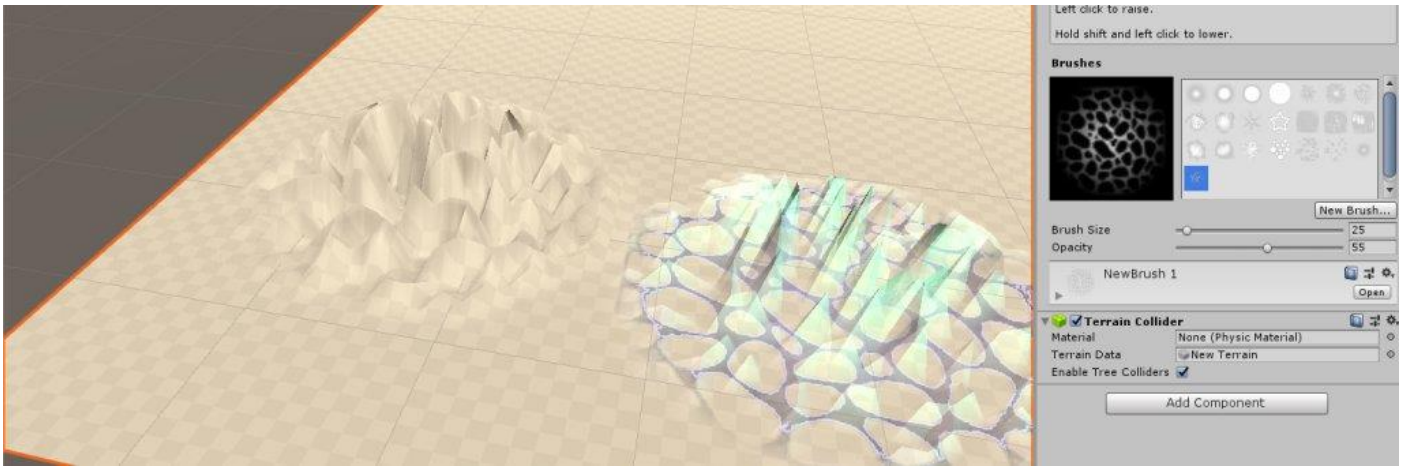
Tip #50: utiliser des brush personnalisées pour vos terrain.

EDITOR

Il est possible d'importer des "brush" personnalisées pour le terrain.Le terrain étant sélectionné, dans l'inspector, faire "*New Brush*" puis sélectionnez la texture souhaitée.



Résultat pour l'outil *Raise or Lower Terrain* (peu s'appliquer aussi par exemple pour *Paint Texture*):

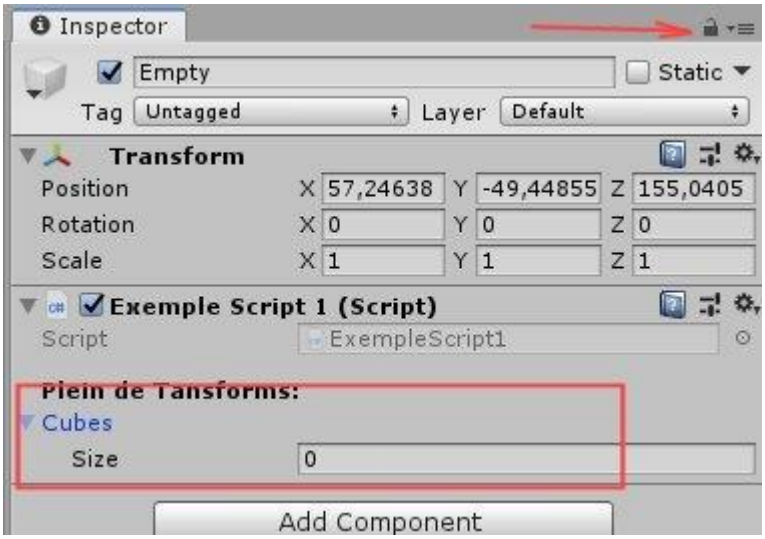


Tip #51: initialiser les tableaux dans l'inspector par bloc.

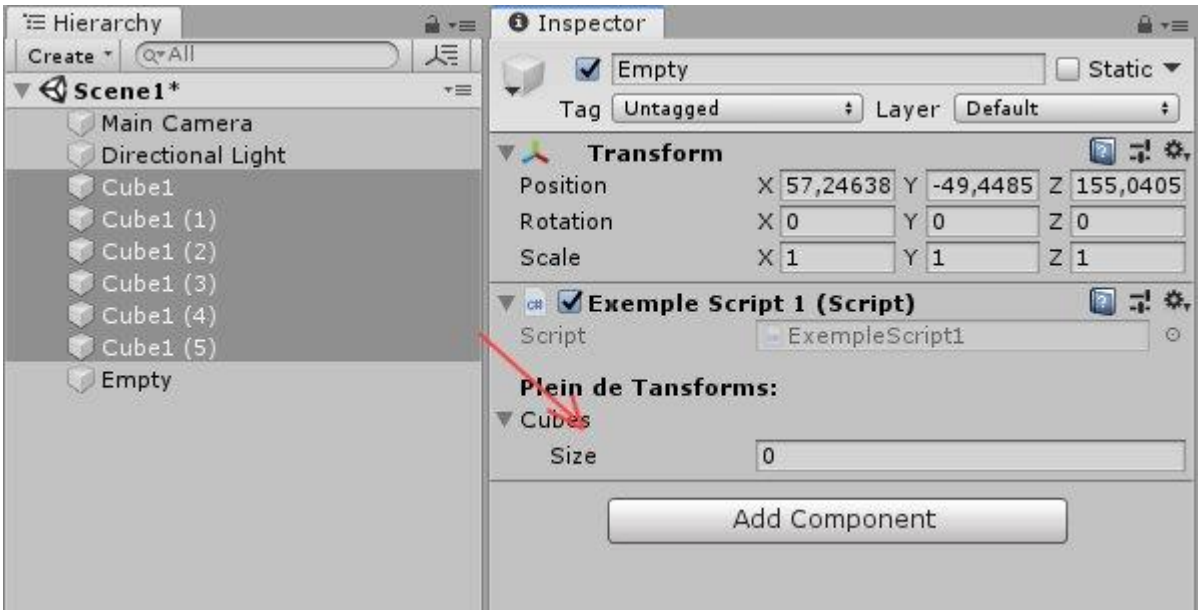
EDITOR

Pour drag and drop une série complète d'éléments pour initialiser un tableau définis et accessible dans l'inspector,

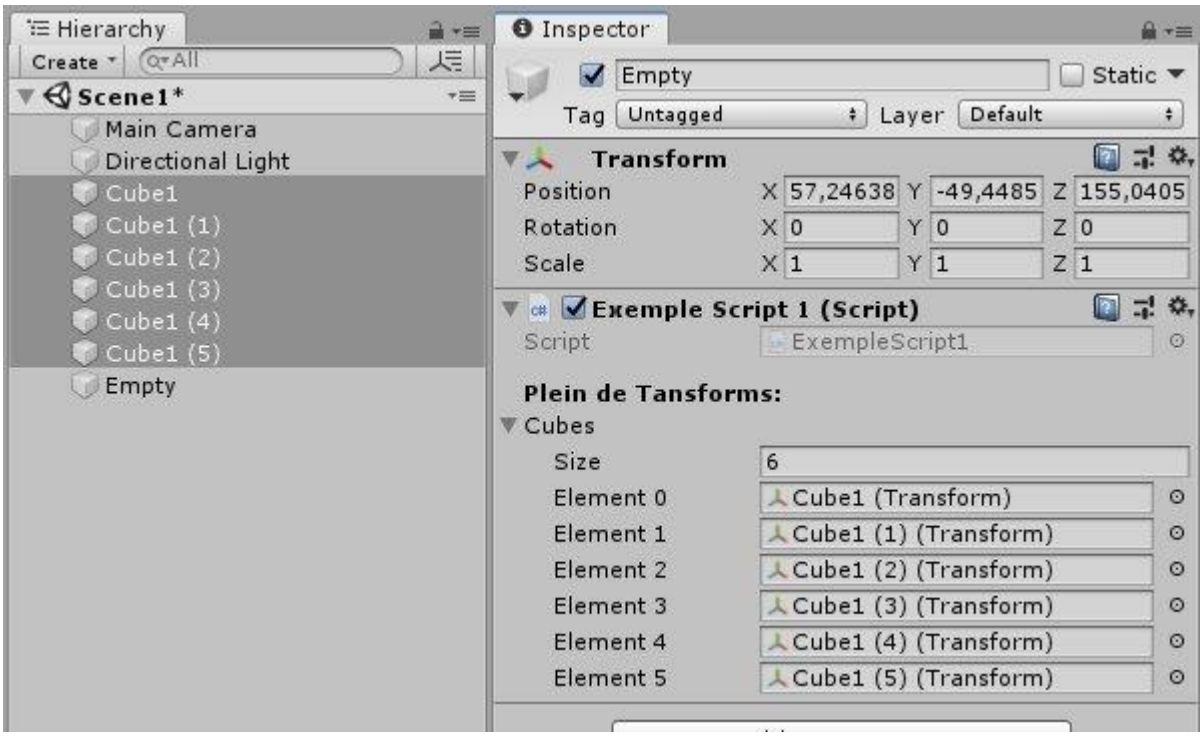
verrouillez l'inspector:



Sélectionnez les objets dans la "*Hierarchy*", puis faire un drag and drop sur le label du tableau cible au sein de l'inspector:



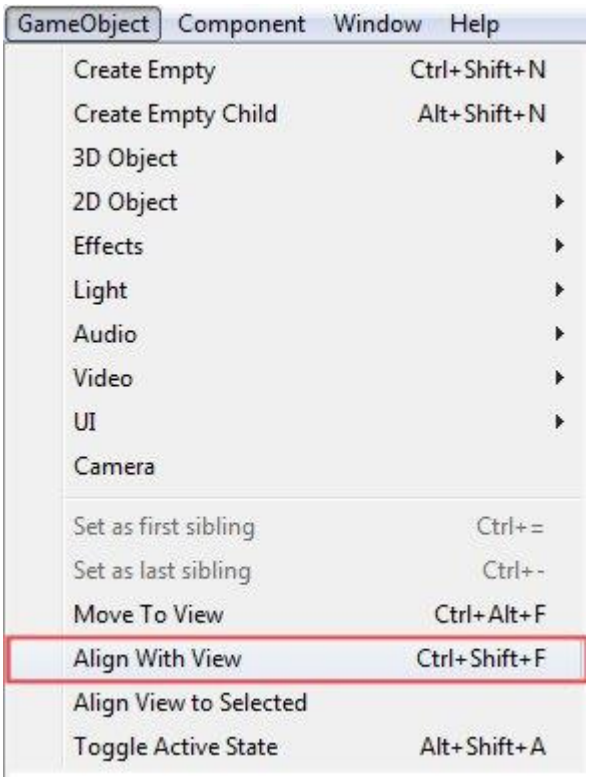
Le tableau est alors initialisé avec l'ensemble des éléments sélectionnés:



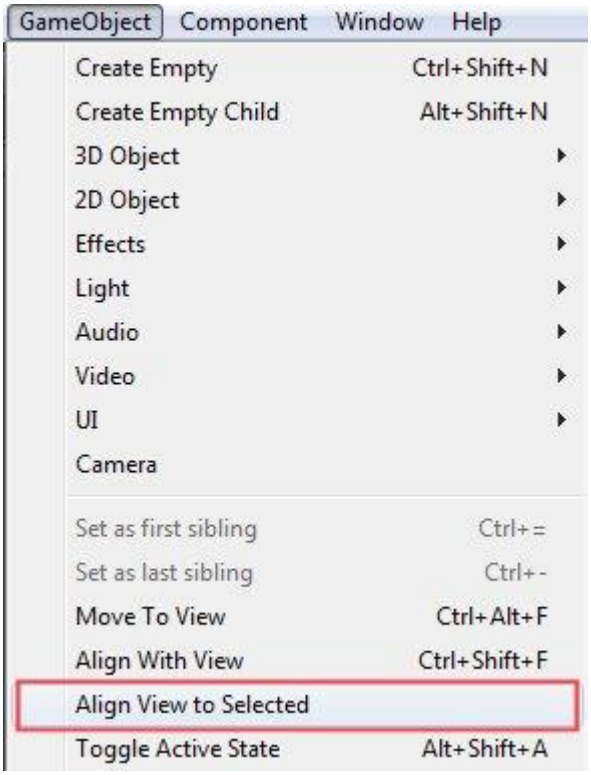
Tip #52: aligner un GameObject sur la vue scène, et inversement.

EDITOR

Pour aligner un *GameObject* (la camera par exemple) sur la vue "scène", l'objet étant sélectionné, au niveau du menu principal, choisissez *GameObject->Align With View*



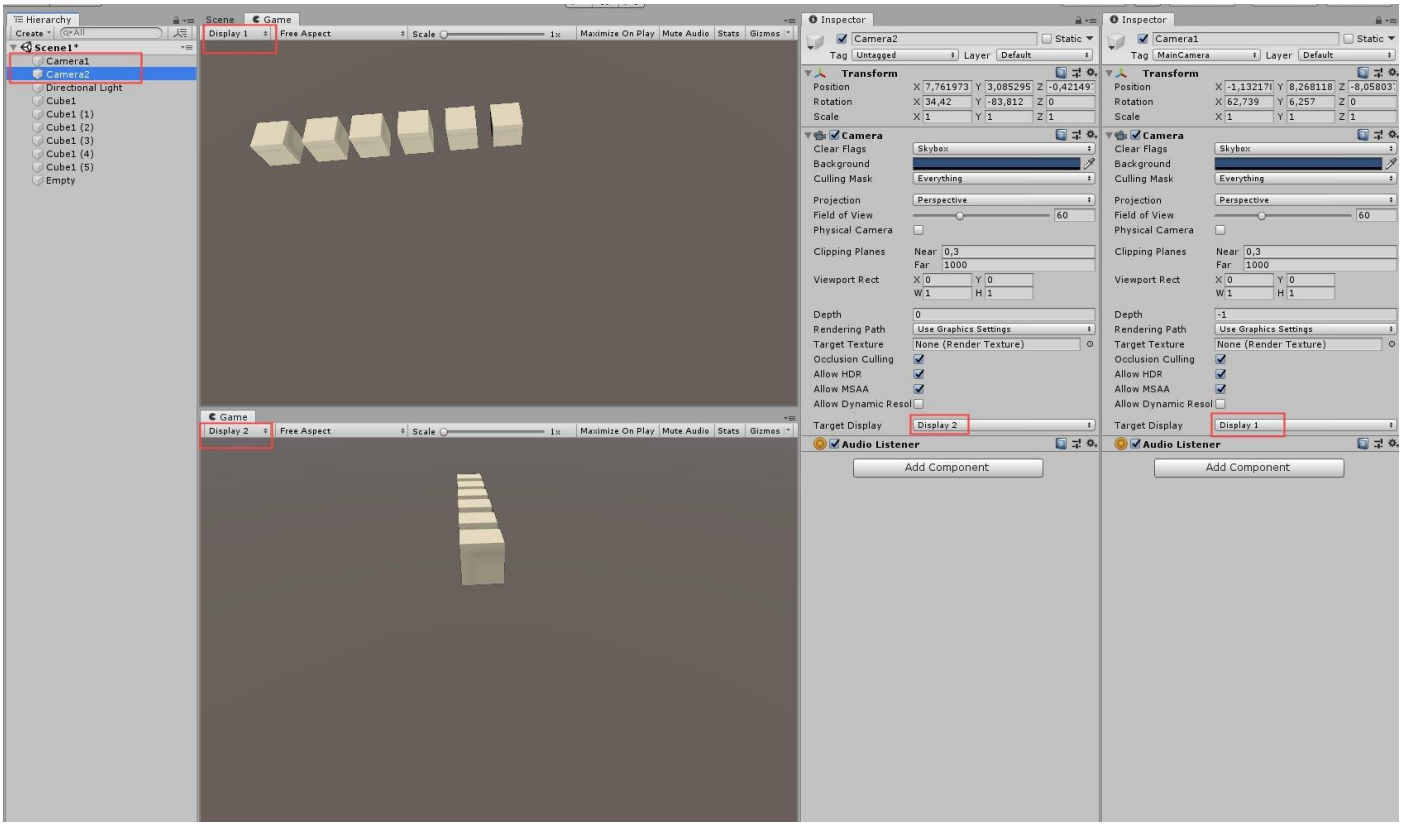
inversement, si vous souhaitez aligner la vue "Scène" par rapport à un GameObject, alors l'objet étant sélectionné, au niveau du menu principal, choisissez *GameObject->Align View to Selected*



Tip #53: simuler le muliti-écran au sein dans l'éditeur.

EDITOR

Utilisez deux cameras pour deux points de vue différents en mode Play:



Lien utile: [Multi-display]

Tip #54: un IEnumerator pour vos Start()

SCRIPTING

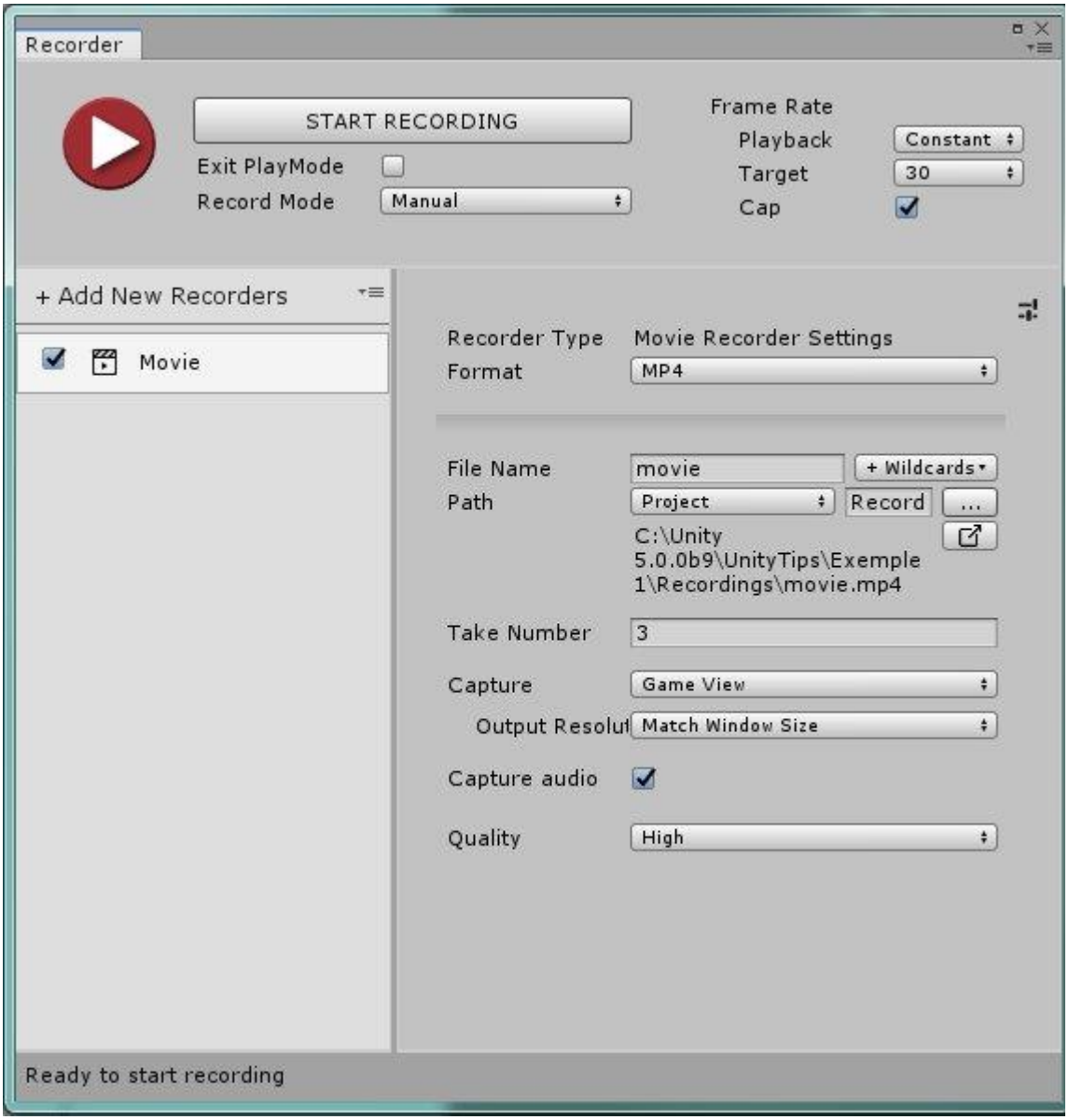
un *IEnumerator* pour vos Start(), pour des séquences de démarrage temporisés:

```
    [SerializeField]
    float tempo1 = 2.0f;
    [SerializeField]
    float tempo2 = 3.0f;

    // Start is called before the first frame update
    private IEnumerator Start()
    {
        Debug.Log("demarrage");
        yield return new WaitForSeconds(tempo1);
        Debug.Log("Action1");
        yield return new WaitForSeconds(tempo2);
        Debug.Log("Action2");
    }
```

Tip #55: enregistrer des vidéos de vos applications.

EDITOR

Enregistrer en vidéo le résultat de vos créations à partir de l'éditeur (en mode Play).
Pour ce faire, sans avoir à utiliser un logiciel tiers (genre Fraps par exemple), vous pouvez le faire directement à partir de l'éditeur grâce à *Unity Recorder*.

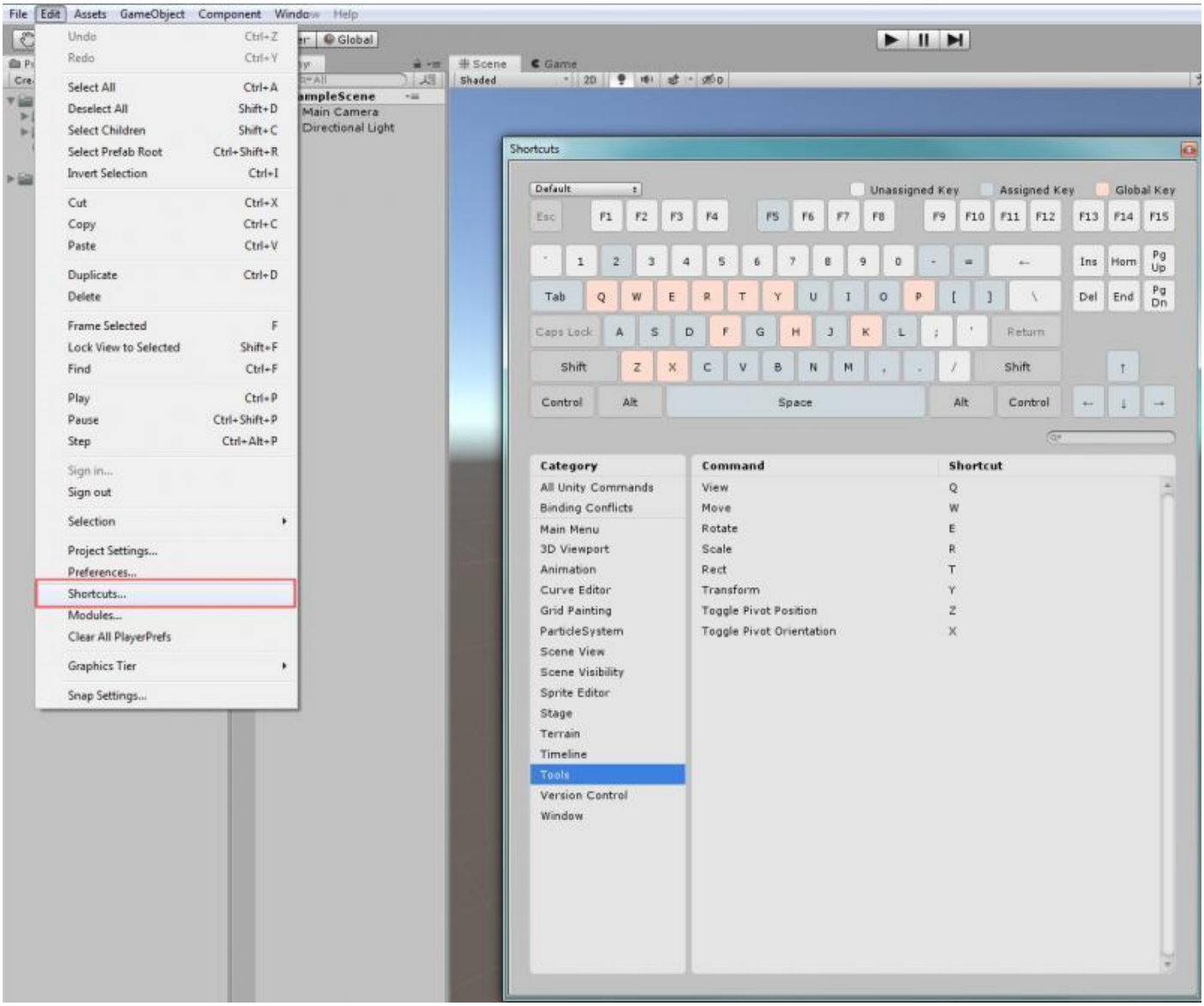


Lien utile: [Unity Recorder]

[Tip #56: choisir ses racourcis clavier.]

EDITOR

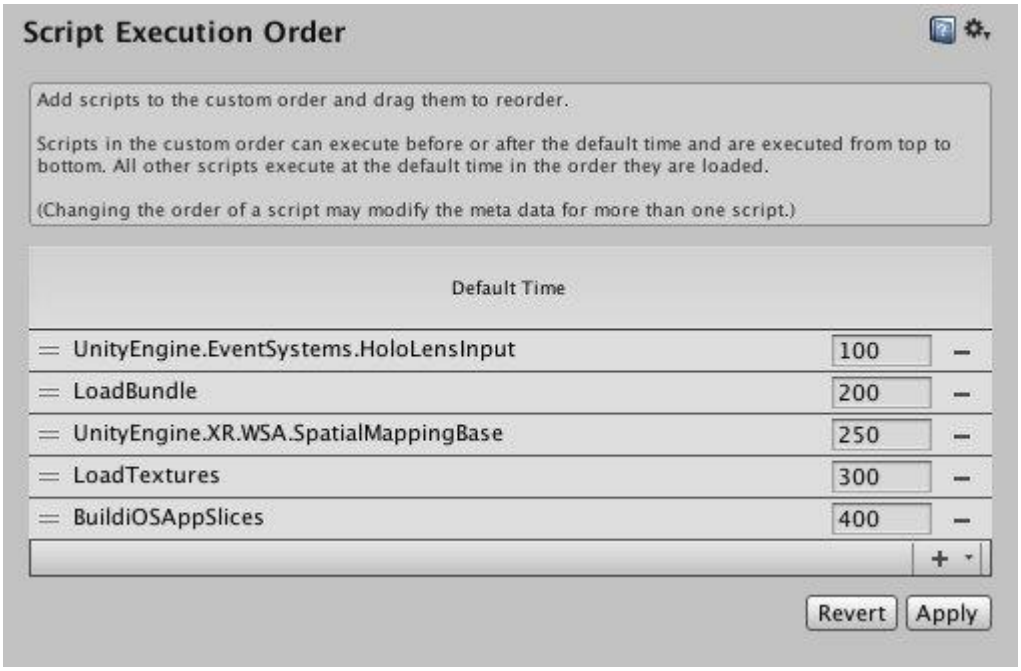
Changez/personnalisez vos raccourcis clavier facilement avec le *shortcuts manager* (Unity 2019):



Tip #57: définir l'ordre d'exécution des scripts.

SCRIPTING

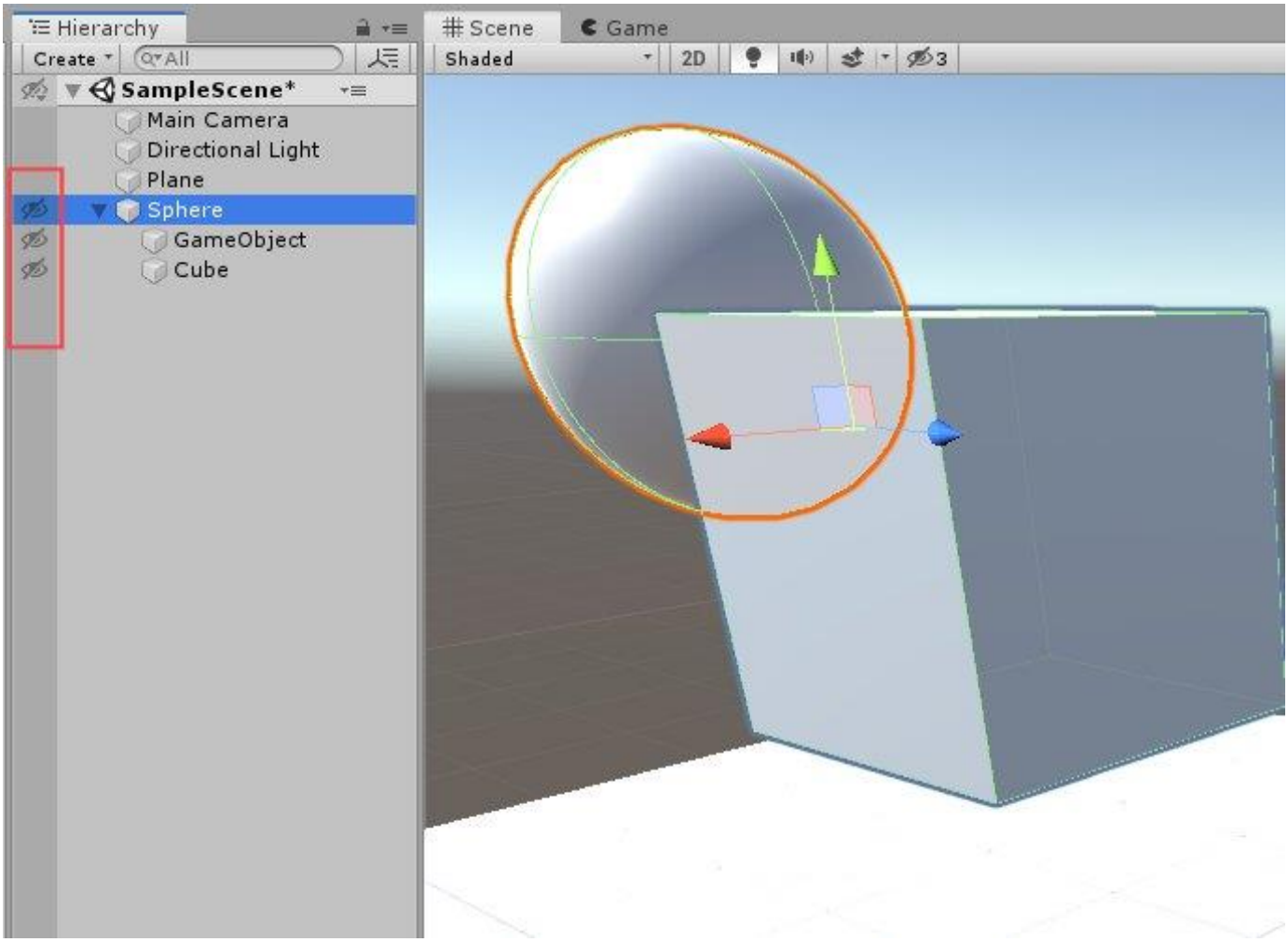
Dans certains cas, il peut être intéressant de pouvoir définir l'ordre d'exécution de certains scripts. Pour ce faire, dans le menu principal, choisir *Edit -> Project Settings*, puis *Script Execution Order*:



Tip #58: masquer et afficher rapidement des objets dans la vue Scène

EDITOR

Quand vous travaillez sur des scènes très chargées, utilisez les commandes *SceneVis* pour masquer et afficher rapidement (ou isoler temporairement) des objets dans la vue Scène, sans modifier la visibilité des objets dans le jeu.

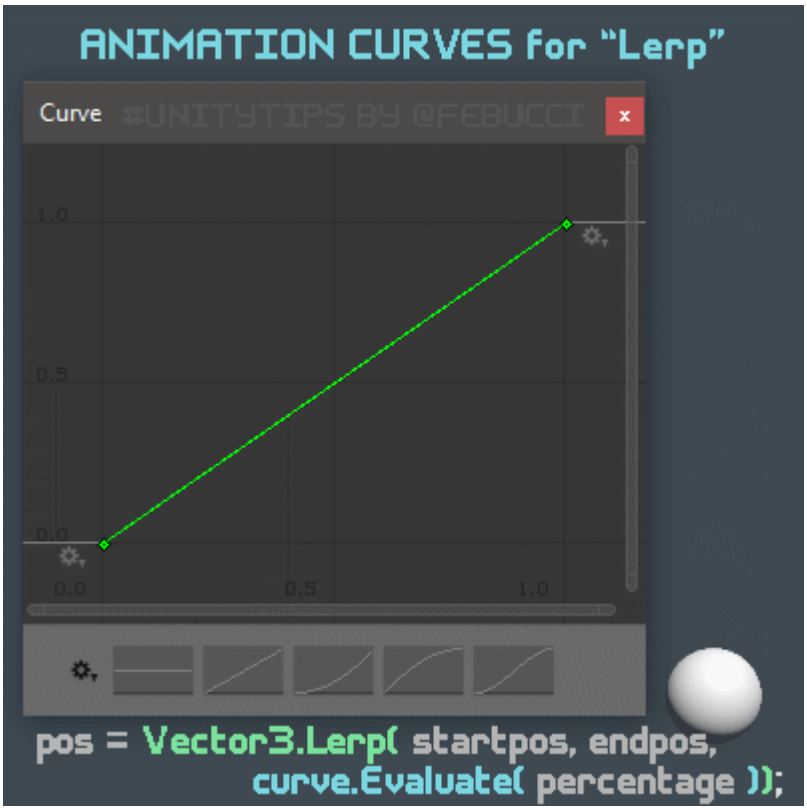


[Tip #59: animation curve pour vos Lerp.]

Bonjour,

SCRIPTING

Utiliser les AnimationCurve pour vos Lerp:




```
using UnityEngine;

public class TestLerpCurve : MonoBehaviour
{
    [SerializeField] private Transform startpos;
    [SerializeField] private Transform endpos;
    [SerializeField] private AnimationCurve curve;
    private float percentage;
    private Vector3 pos;
    private float dir = 1;

    private void Update()
    {
        pos = Vector3.Lerp(startpos.position, endpos.position,
curve.Evaluate(percentage));
        transform.position = pos;

        percentage += Time.deltaTime * dir;
        if (percentage > 1.0f || percentage < 0)        { dir = -dir; }
    }
}
```

Tip #60: la méthode Reset pour réinitialiser vos composants.

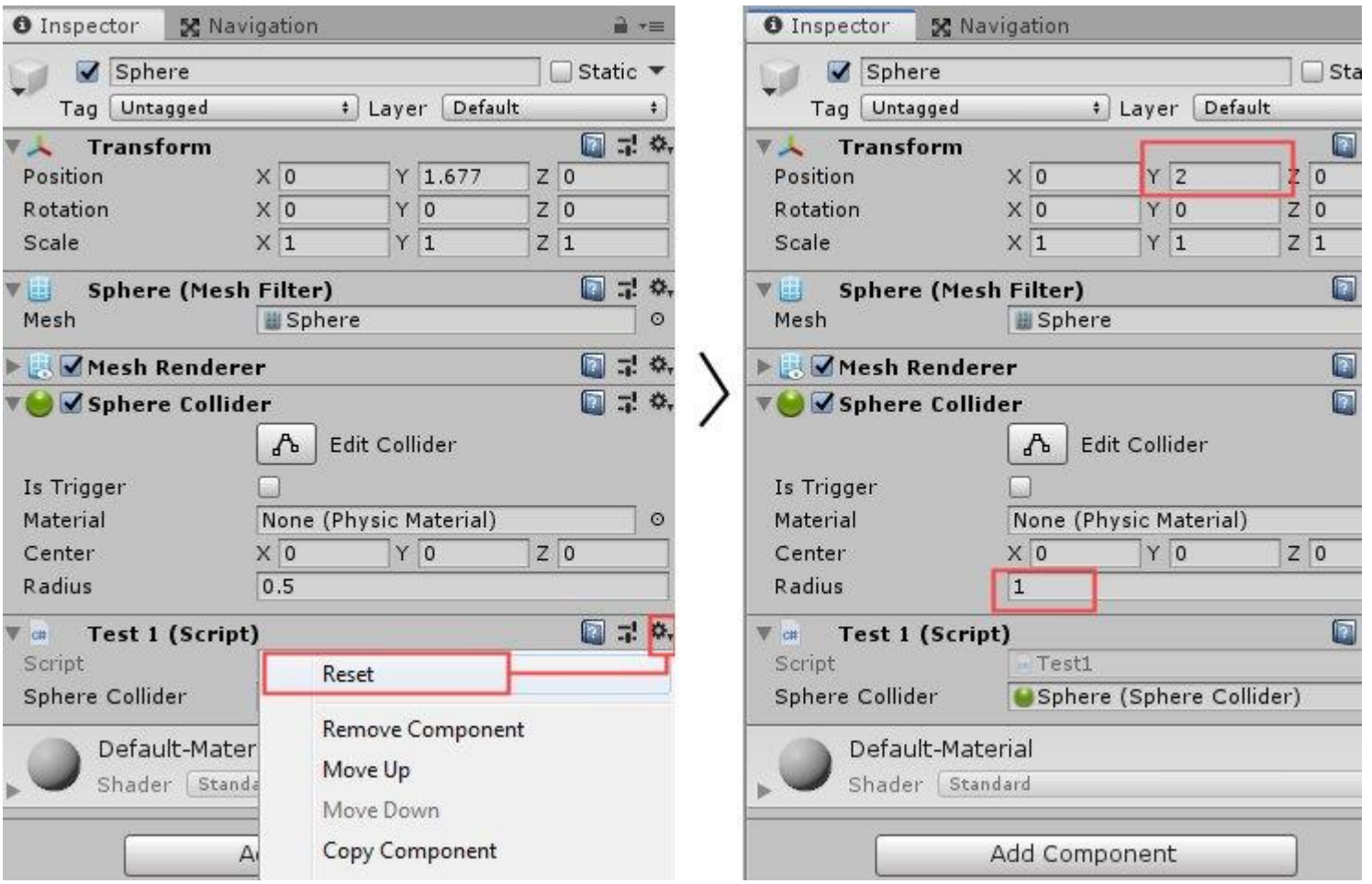
Bonjour,

SCRIPTING

Le *Reset* d'un script au niveau de l'inspector est lié à la méthode *Reset()*, permettant ainsi une réinitialisation de certains éléments des autres components du GameObject.

Exemple:

```
using UnityEngine;

[RequireComponent( typeof(SphereCollider))]
public class Test1 : MonoBehaviour
{
    [SerializeField] private SphereCollider sphereCollider;
    ....
#if UNITY_EDITOR
    private void Reset()
    {
        sphereCollider  = GetComponent<SphereCollider>();
        sphereCollider.radius = 1.0f;
        transform.position = new Vector3(0, 2.0f, 0);
    }
#endif
}
```



Tip #61: contrôler la valeur des données entrées dans l'inspector.

Bonjour,

SCRIPTING

Il est possible d'opérer un contrôle des valeurs entrées dans les champs d'un composant script au sein de l'inspector,
grâce à l'emploi de la méthode *OnValidate()*.

Exemple:

```
    public int ivar1 = 20;
    [SerializeField]
    private float fvar2 = 1.0f;

#if UNITY_EDITOR
    private void OnValidate()
    {
      if(ivar1<0)
      {
            ivar1 = 0;
      }
      if(fvar2>10.0f)
      {
            Debug.Log("Attention, dépassement de valeur");
            fvar2 = 10.0f;
      }
    }
#endif
```

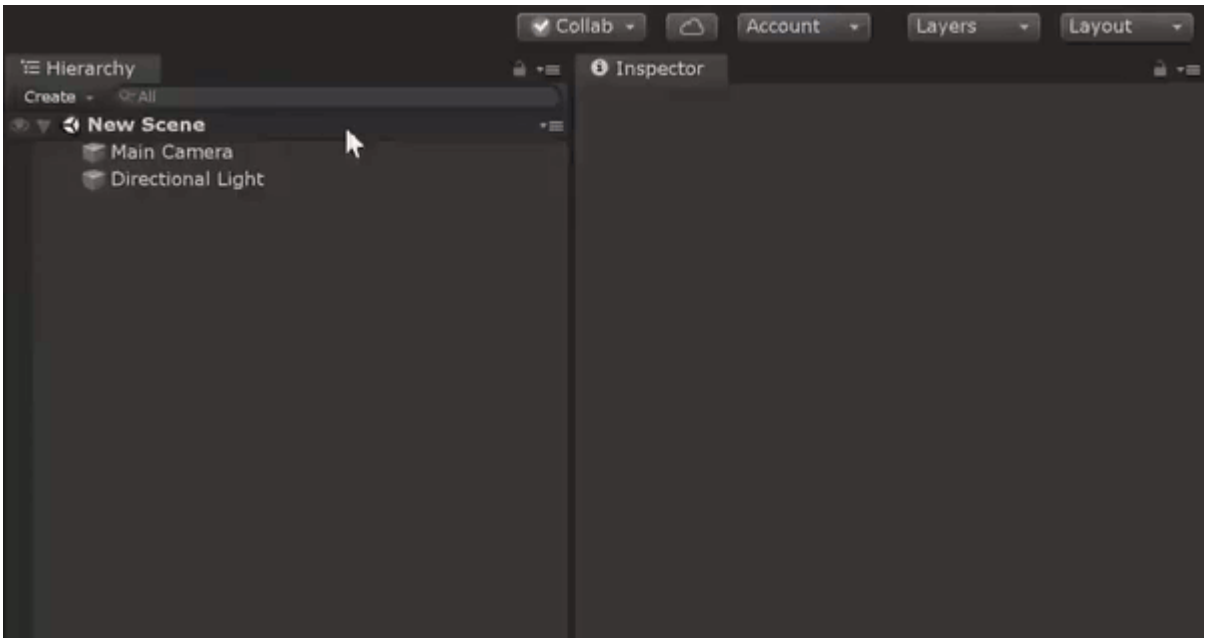
Lien Utile**:** **OnValidate()**

Tip #62: structurer les Tags.

Bonjour,

EDITOR

Si vous utilisez beaucoup de Tags dans vos projets, vous pouvez les regrouper en ajoutant une barre oblique (/) dans le nom du tag pour créer une arborescence semblable à une structure.



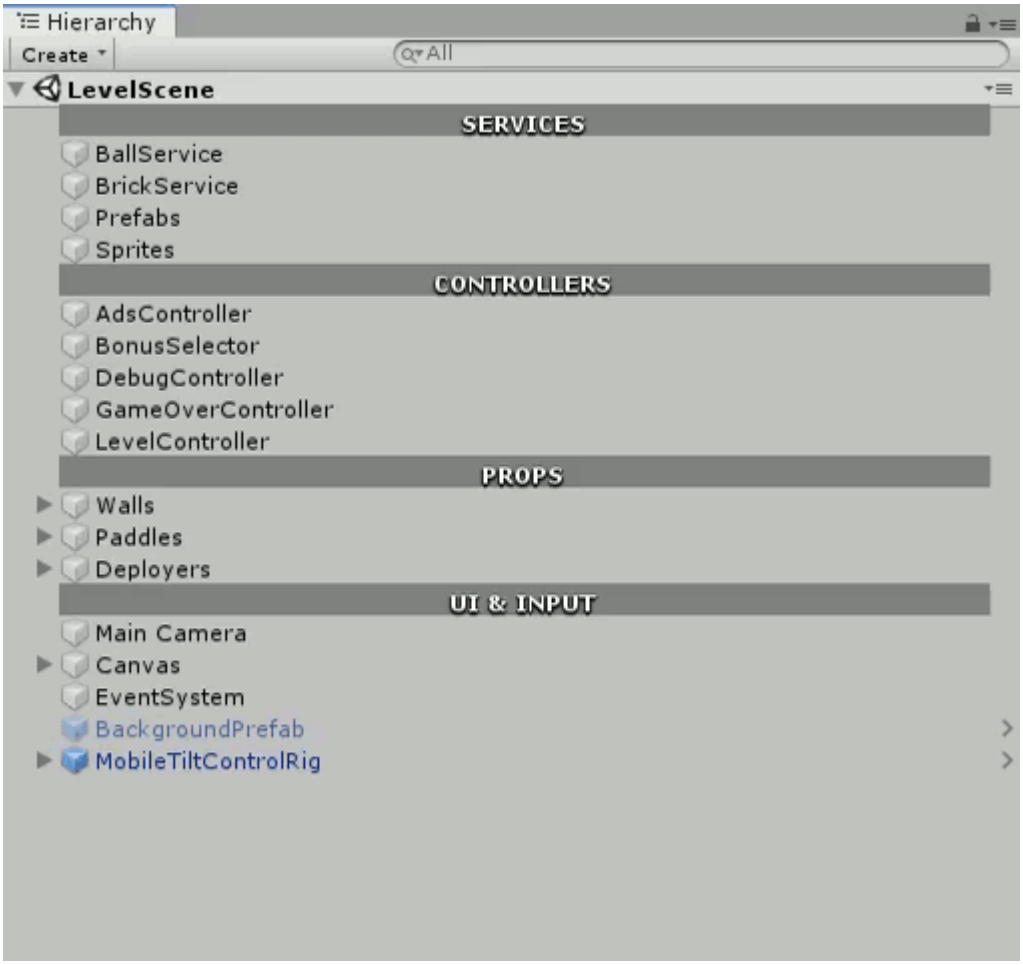
Tip #63: améliorer la visibilité des séparateurs dans la hierarchy,

Bonjour,

EDITOR

Dans le même esprit que ce qui avait été décrit dans le Tip n°15 sur l'organisation des GameObjects dans la hierarchy, *hierarchyWindowItemOnGUI* permet d'améliorer (entre autre) la visibilité des séparateurs:

```
using UnityEngine;
using UnityEditor;

[InitializeOnLoad]
public class HierarchyWindowGroupHeader : Editor
{
    static HierarchyWindowGroupHeader()
    {
        EditorApplication.hierarchyWindowItemOnGUI +=
HierarchyWindowItemOnGUI;
    }

    static void HierarchyWindowItemOnGUI(int instanceID, Rect
selectionRect)
    {
        var gameObject = EditorUtility.InstanceIDToObject(instanceID) as
GameObject;
        if (gameObject != null && gameObject.name.StartsWith("---",
System.StringComparison.Ordinal))
        {
            EditorGUI.DrawRect(selectionRect, Color.gray);
            EditorGUI.DropShadowLabel(selectionRect,
gameObject.name.Replace("-", "").ToUpperInvariant());
        }
    }
}
```



Source: http://diegogiacomelli.com.br
Lien utile: hierarchyWindowItemOnGUI

Tip #64: provoquer, par script, la mise en pause d'une application dans l'éditeur.

Bonjour,

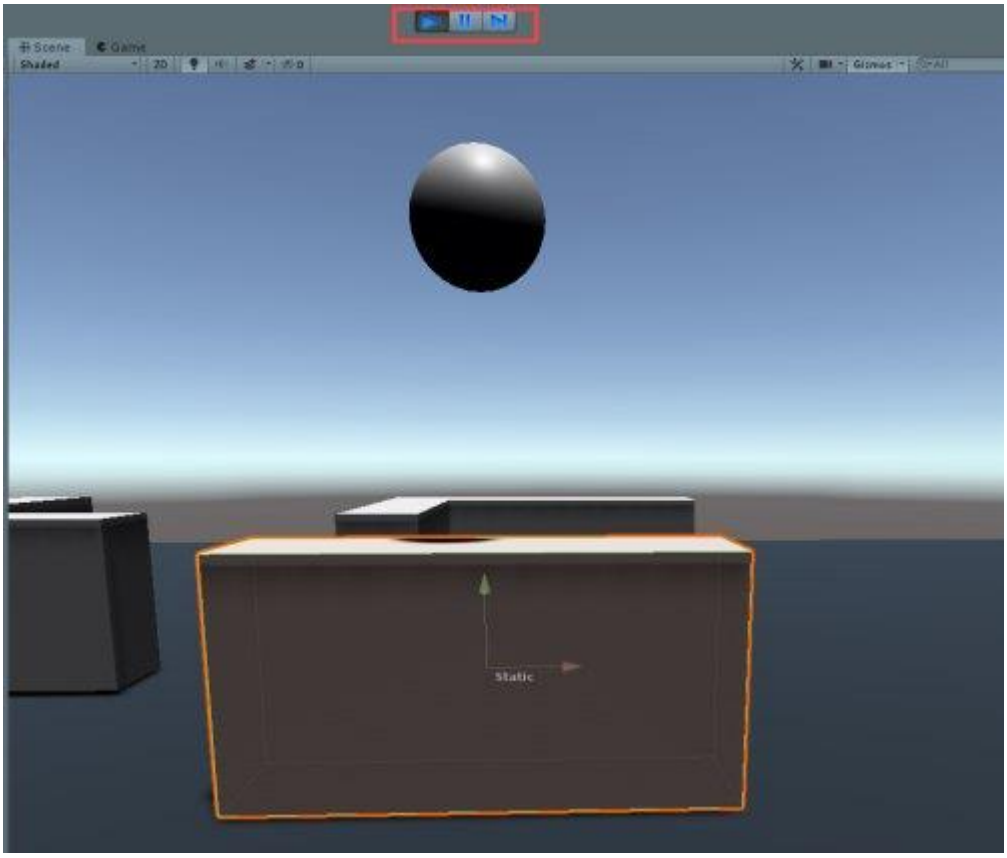
SCRIPTING EDITOR

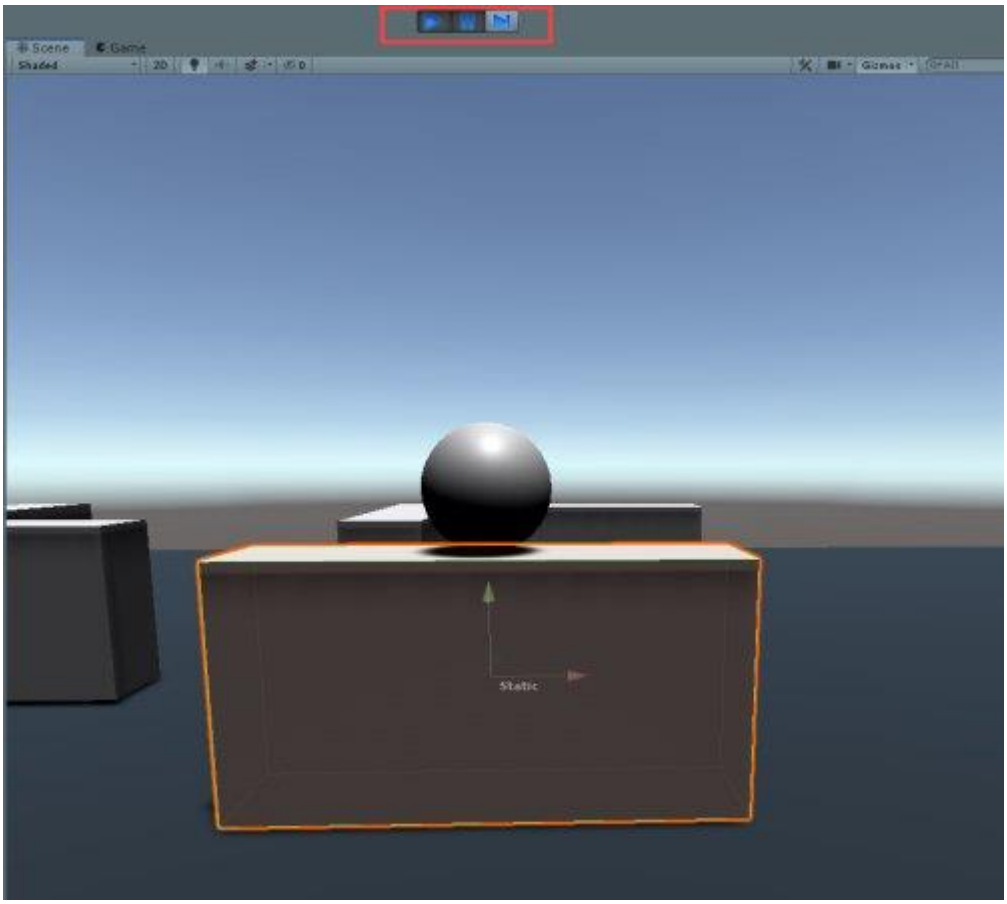
Il est possible de provoquer, par script en runtime, la mise en pause de l'application en cours d'exécution au sein de l'éditeur.
Exemple d'utilisation lors d'une détection de collision:

emple d'utilisation lors d'une détection de collision:

Code:

```
private void OnCollisionEnter(Collision collision)
{
    Debug.Log("OnCollisionEnter");

    Debug.Break();
}
```


Scene
Game
Shaded
2D
Gizmos
Static


Scene
Game
Shaded
2D
Gizmos
Static

Tip #65: afficher en mode Play les sorties console directement dans la fenêtre Game.

Bonjour,

SCRIPTING

Grâce à Application.logMessageReceived, il est possible d'afficher directement les sorties console dans la fenêtre Game en mode Play.

Exemple simple d'utilisation:

```
using UnityEngine;

public class TestLog : MonoBehaviour
{
    private string tips = null;

    private void Awake()
    {
        Application.logMessageReceived += ShowTips;
    }

    private void Update()
    {
        if(Input.GetKeyDown(KeyCode.Space))
        {
            Debug.Log("Exemple de message");
        }
    }

    private void OnApplicationQuit()
    {
        Application.logMessageReceived -= ShowTips;
    }

    private void ShowTips(string msg, string stackTrace, LogType type)
    {
        switch (type)
        {
            case LogType.Error:
                tips += "<color=red>" + msg + "</color>" + "\r\n";
                break;
            case LogType.Assert:
                tips += msg + "\r\n";
                break;
            case LogType.Warning:
                tips += "<color=yellow>" + msg + "</color>" + "\r\n";
                break;
            case LogType.Log:
                tips += msg + "\r\n";
                break;
            case LogType.Exception:
                tips += "<color=red>" + msg + "</color>" + "\r\n";
                break;
            default:
                break;
        }
    }

    void OnGUI()
    {
        GUI.Label(new Rect(16, 16, Screen.width / 2, Screen.height),
tips);
    }
}
```

Exemple d'utilisation: