Jxx – Outils – Outils Graphique

Modification Dans Unity

Gizmos

Utilité : aides visuelles de débogage ou de configuration dans la vue Scène.

Plusieurs facons de les ajouter dans la View :

- Redefinir OnDrawGizmos ou OnDrawGizmosSelected

```
Gizmos.color = Color.blue;
Gizmos.DrawWireCube(transform.position, new Vector3(4, 4, 4));
```

- Utiliser les attributs [SomeAttribut]

```
public class MesGuizmo{
    [DrawGizmo(GizmoType.Selected | GizmoType.Active)]
    private static void MyCustomOnDrawGizmos(PlayerScript playerScript, GizmoType gizmoType) {
        Gizmos.color = Color.blue;
        Gizmos.DrawCube(playerScript.transform.position, new Vector3(4, 4, 4));
    }
}
```

- Dans les script Editor (Plus tard dans le cours) en utilisant la class Handles

- Grace à la classe Log

```
Log.DrawLine
```

Gizmos

Exo :

Tracer des lignes entre des GameObjects present dans une List<GameObject>

Inspector / Attributs

- Par defaut tous les champs public sont serializé et visible dans l'inspector
- Attributs Spécifiques
  - [Serializable] pour une class ou une struct
  - [SerializeField] permet de rendre un attribut prive visible dans l inspector
  - [HideInInspector] permet de cacher un champ public
  - [RequireComponent(typeof(RigidBody)] surtout pour éviter les erreurs force a avoir cet attribut en composition
  - [ExecuteInEditMode] Cet attribut permet à la fonction Update d'etre appelle meme en edit mode

https://docs.unity3d.com/ScriptReference/ExecuteInEditMode.html

Custom Inspector

- Doit mettre le script dans un répertoire **Editor**
- Doit herite de Editor
- Doit Utiliser l attribut [CustomEditor(typeof(Class Herite de MonoBehavior))]
- Peut ajouter [CanEditMultipleObjects] si besoin que l inspector soit sur plusieurs GameObject.

- Existe la variable target → reference vers l objet ou on redéfinit l editor
- Si sur plusieurs object on aura targets
- Comme Herite de Editor on pourra redefinir
  - **OnEnable**
  - **OnDisable**
  - **OnDestroy**
  - **OnValidate**
  - **OnInspectorGUI**

OnInspectoGUI

- DrawDefaultInspector()
- Classe pour dessiner des elements dans inspector
  - GUI
  - GUILayout
  - GUILayoutUtility
  - EditorGUILayout
  - EditorGUI
  - EditorStyles
  - GUILayoutOption
- Allez voir la doc et stackoverflow …
- https://answers.unity.com/questions/601131/editorgui-editorguilayout-gui-guilayout-pshhh-when.html
- Event sur les composants: retournent une valeur ou un bool (cas du bouton)

#

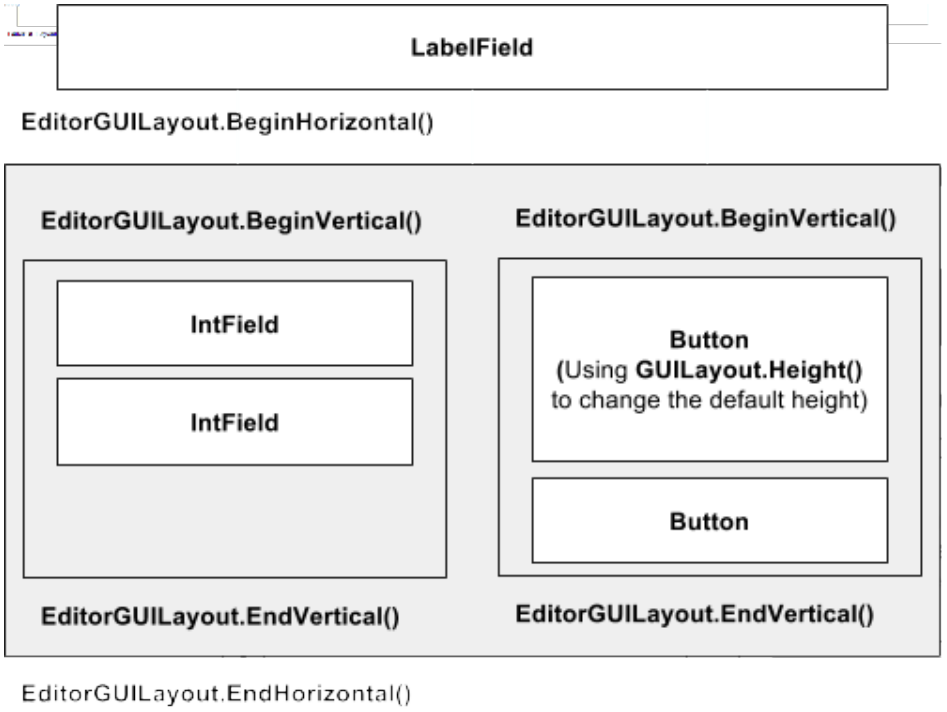
Methodes Utiles / Layout

- Methodes

```
if(GUI.changed) {
    EditorUtility.SetDirty(_myTarget);
}
```

GUI.color permet de connaître ou setter la couleur a utiliser
GUI.enable getter/setter
EditorGUIUtility.currentViewWidth
EditorGUIUtility.singleLineHeight

- Layout Auto



Customiser La Scene View

- les gizmos
- Depuis l Editor en redéfinissant OnSceneGUI.
  - Peut redessiner des Gizmos mais aussi des composants comme un bouton
  - Utilise la class Handles
  - Pour des composants, comme le bouton , doit etre entre Handles.BeginGUI() et Handles.EndGUI()
  - HandleUtility.GUIPointToWorldRay et HandleUtility.WorldToGUIPoint
  - GUILayout.BeginArea(rect)
- Class Tool manipule les outils de deplacement scale rotation
  - Tools.current
- SceneView.currentDrawingSceneView.in2DMode = true;

Force le mode 2D dans unity

Event

- Peut catégoriser le type d événement
  - https://docs.unity3d.com/ScriptReference/EventType.html
  - https://docs.unity3d.com/ScriptReference/Event.html

Vector2 position = Event.current.mousePosition

- Événement Spécial le DragAndDrop

Deux types a verifier au minimum

  - EventType.DragPerform
  - EventType.DragUpdated

1) Ajouter event.Use() pour consumer les evenements (Event.PopEvent)
2) Dans dragupdate ajouter cette ligne :

```
DragAndDrop.visualMode = DragAndDropVisualMode.Copy
```

3) Dans dragperform

```
DragAndDrop.AcceptDrag()
foreach (UnityEngine.Object dragobj in DragAndDrop.objectReferences) {
        GameObject go = dragobj as GameObject;
        if (!go) {
         Debug.Log("Drop Fail");
        }
}
```