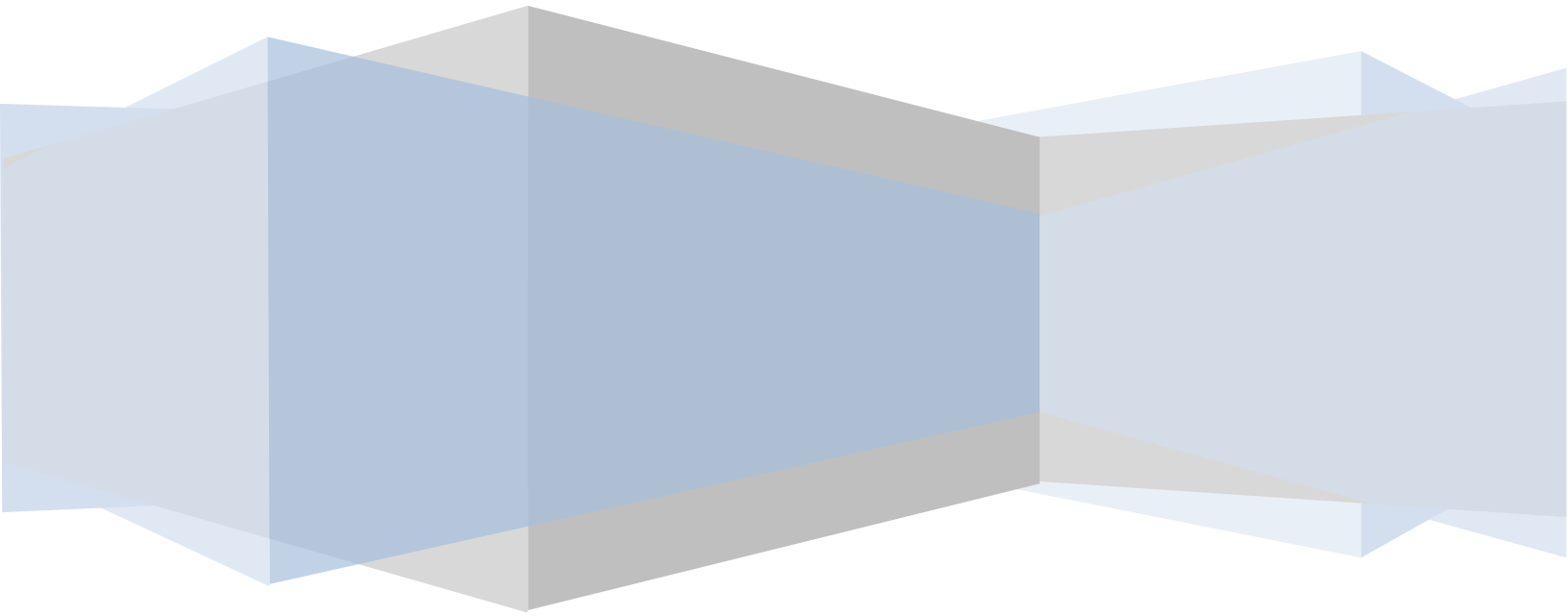
BInspector v1.3 Manual

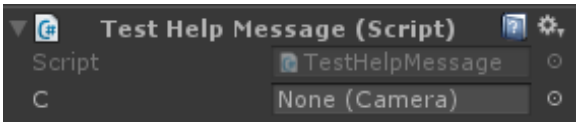
Wayne Su



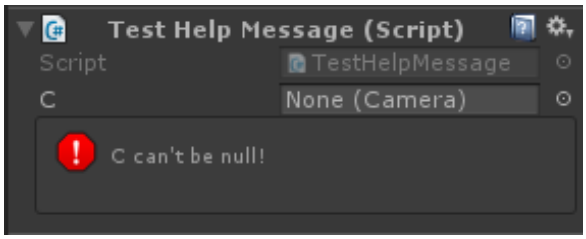
Contents

Introduction

本專案目的在於優化 Unity Inspector，提升遊戲開發進度。由於 Inspector 原始介面較為簡陋，有些參數限制只能在遊戲執行時才能判斷是否合乎規範；如圖表 1，此變數並無提醒若為空值是否會造成遊戲錯誤，只能透過程式碼在遊戲執行時檢查；若能在編輯數據時告知開發人員此錯誤，即能避免不必要的 Debug 時間，以便提升專案開發進度(圖表 2)。

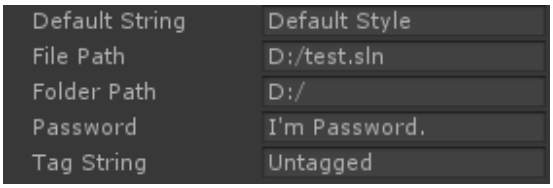


圖表 1. 原介面



圖表 2. 空值則出現警告訊息

此外，本專案改善部分變數的顯示方式，以便開發者編輯與閱讀。



圖表 3. string 原始顯示方式

圖表 4. string 依照模式不同改善顯示方式

備註：專案內有範例場景可供實際操作練習

Change Log

v1.3:

- Add “Conditional Hide”

v1.2:

- Fix “Reorderable List” crash bug
- Change “Reorderable List” interface
- “Instead Show By Property Drawer” can be multiple edited
- “Array Element Title” can use enum be the display name

v1.1:

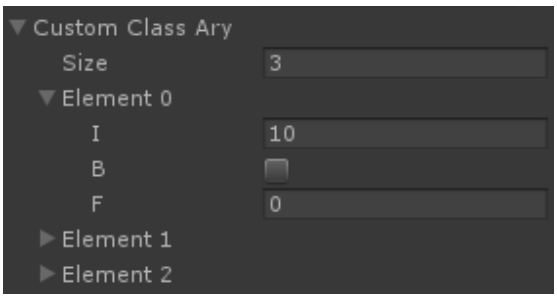
- Add “Scriptable Object Create Button”, “Scriptable Object File Location”, “Sub View” Inspector
- Fix property height
- Add “Category” mode in “String Style”

v1.0:

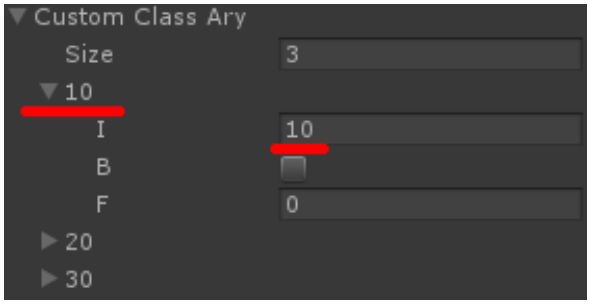
- Add “Array Element Title”, “Descriptor”, “Enum Mask”, “Help Message”, “Instead By Property”, “Range”, “Reorderable List”, “String Style” Drawer

Array Element Title

目的: 使用 class 某一變數當作顯示名稱，而非 Element 0, 1, 2 …



圖表 5. List 原始顯示方式



圖表 6. 依據某變數值做為顯示名稱

使用方法 1:

```
// Parameters:
//   typeVarName: 欲當作顯示名稱的變數名稱


/// <summary>
/// Display name will use "i" value
/// </summary>
[BInspector.ArrayElementTitle("i")]

public CustomClass[] customClassAry = new CustomClass[0];
```

```
[Serializable]
public class CustomClass
{
     public int i = -1;
     public bool b = false;
     public float f = 0f;

}
```

使用方法 2:

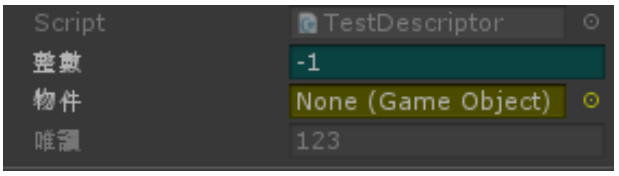
```
// Parameters:
//   enumType: 欲當作顯示名稱的 Enum


/// <summary>
/// Display name will use enum value
/// </summary>
[BInspector.ArrayElementTitle(typeof(TestEnumTitle))]

public CustomClass[] customClassAry2 = new CustomClass[0];
```

```
public enum TestEnumTitle
{
    Enum1,
    Enum2,
    Enum3,
}
```

Descriptor

目的: 更改變數顯示名稱、提示、顏色、唯讀

圖表 7. 原變數顯示方式



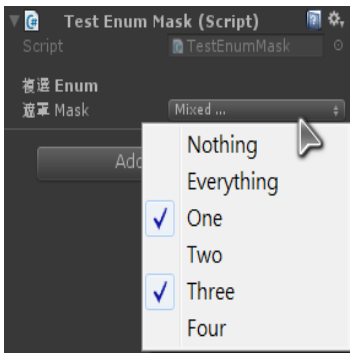
圖表 8. 可變動的名稱、背景色、唯讀模式

使用方法:

```
// Parameters:
//   name: 顯示名稱
//   tips: 註解 (可不填)
//   r, g, b, a: 背景色 (可不填)
//   isReadOnly: 是否唯讀 (預設為false)
[BInspector.Descriptor("整數", "I'm tooltip", 0, 1, 1, 1)]
public int testInt = -1;

[BInspector.Descriptor("物件", "請拉物件至此", 1, 1, 0, 1)]
public GameObject testGo = null;

[BInspector.Descriptor("唯讀", null, true)]
public float testFloat = 123f;
```

Enum Mask

目的: 將 enum 使用 mask 顯示 (即可複選)



圖表 9. enum 變為 mask 顯示方式

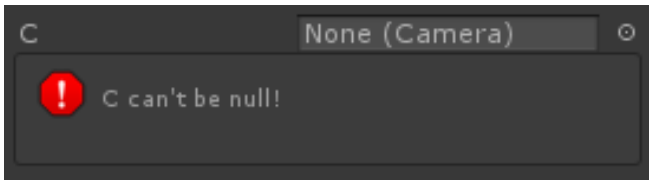
使用方法:

```
// Parameters:
//   displayName: 顯示名稱 (不填即使用原本名稱)
[BInspector.EnumMask("遮罩 Mask")]
public MyMask myMask;
```

```
// enum 數值必須按照: 1, 2, 4, 8, 16, 32, 64, etc. 否則取值會造成錯誤
// 不需設定 數值0 的選項
public enum MyMask
{
    One = 1,
    Two = 2,
    Three = 4,
    Four = 8,
}
```

Help Message

目的: 依照條件判斷顯示 一般/警告/錯誤 訊息



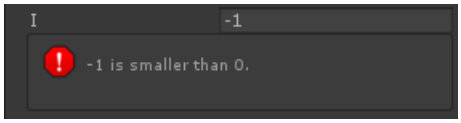
圖表 10. 依照開發者設定，決定出現警告訊息的條件

使用方法 1: 使用內建判斷式 Common Judge Type

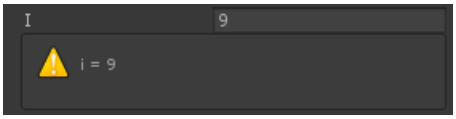
```
// Common Judge Type 為常見判斷式. 條件成立則顯示提示訊息:
//   IntNegative: 整數為負數
//   IntZero: 整數為 0
//   FloatNegative: 浮點數為負數
//   StringNullOrEmpty: 字串為 null
//   ReferenceNull: 物件為 null

[BInspector.HelpMessage(CommonJudgeType.ReferenceNull)]
public Camera c = null;
```

使用方法 2: 使用開發者自定義函式判斷

```
// 使用自定義判斷函式
// Parameters:
//   CustomMethodName: 欲使用之判斷函式名稱

[BInspector.HelpMessage("Show")]
public int i = -1;


HelpMessageContent helpMsgContent = new HelpMessageContent();
public HelpMessageContent Show()
{
    if (i < 0)
    {
        helpMsgContent.SetMessage(HelpType.Error, i + " is smaller than 0.");
        return helpMsgContent;
    }
    else if (i >= 0 && i < 10)
    {
        helpMsgContent.SetMessage(HelpType.Warning, "i = " + i);
        return helpMsgContent;
    }
    else return null;
}
```



圖表 11. 開發者自定義條件 i < 0



圖表 12. 開發者自定義條件 i >=0 && i < 10

Instead By Property

目的: 利用 Property (getter, setter) 取代顯示變數

圖表 13. 原變數 b 改為顯示 BoolProperty

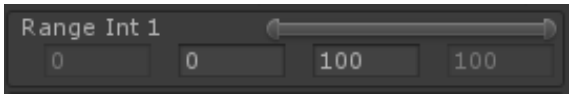
圖表 14. 編輯變數時，是直接執行 Property

使用方法:

```
// (注意) 必須搭配 SerializeField 使用
// Parameters:
//   propertyName: 欲顯示的 Property 名稱
//   displayName: 欲顯示的名稱 (預設為使用 Property 原本名稱)
[BInspector.InsteadShowByProperty("BoolProperty", "Bool Prop"), SerializeField]
bool b = false;


public bool BoolProperty {
    get { return b; }
    set {
        Debug.Log("Set b: " + value);
        b = value;
    }
}
```

Range

目的: 顯示 Min Max Slider



圖表 15. 左右極值不能編輯

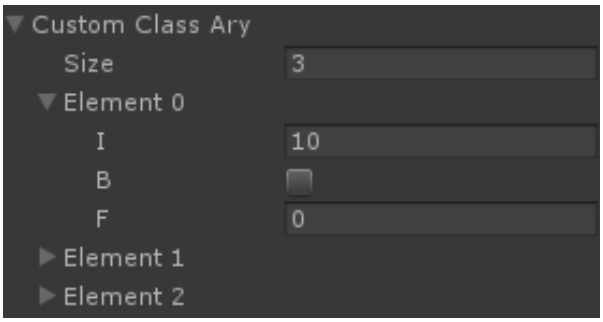
圖表 16. 可編輯左右極值

使用方法:

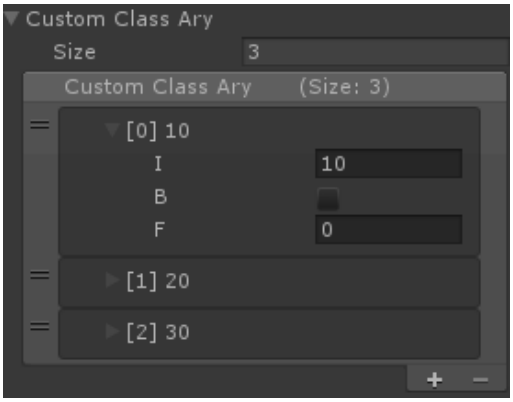
```
// 分為 RangeInt 以及 RangeFloat
// Parameters:
//   minLimit: 極小值
//   maxLimit: 極大值
//   defaultMinVal: 預設較小值
//   defaultMaxVal: 預設較大值
//   isFixedRange: 是否限制修改極大極小值 (預設為True)

public RangeInt rangeInt1 = new RangeInt(0, 100);
public RangeInt rangeInt2 = new RangeInt(0, 100, 25, 75, false);
public RangeFloat rangeFloat1 = new RangeFloat(-10f, 10f, false);
public RangeFloat rangeFloat2 = new RangeFloat(-10f, 10f, -5f, 5f);
```

Reorderable List

目的: 可調順序之清單



圖表 17. List 原始顯示方式



圖表 18. 可改順序之清單介面，並可更改顯示名稱

使用方法:

```
// Parameters:
//   elementTitleVar: 欲當作顯示名稱的變數名稱


/// <summary>
/// Display name will use "i" value
/// </summary>
[BInspector.Reorderable("i")]

public CustomClass[] customClassAry = new CustomClass[0];
```

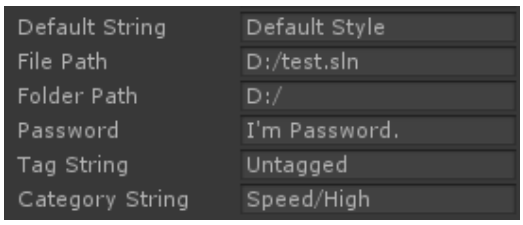
```
[Serializable]
public class CustomClass
{
    public int i = -1;
    public bool b = false;
    public float f = 0f;
}
```

String Style

目的: 更改 string 顯示方式

使用方法:

```
// String Style Type 為顯示字串的模式:
//   Default: 預設樣式. 與原介面無差異
//   FilePath: 搜尋檔案模式. 會開啟搜尋檔案視窗
//   FolderPath: 搜尋資料夾模式. 會開啟搜尋資料夾視窗
//   Password: 密文模式
//   Tag 與 Category: Tag 以及自定義下拉選單模式
[StringStyle(StringStyleType.Default)]
public string defaultString = "Default Style";

[StringStyle(StringStyleType.FilePath)]
public string filePath = "D:/test.sln";

[StringStyle(StringStyleType.FolderPath)]
public string folderPath = "D:/";

[StringStyle(StringStyleType.Password)]
public string password = "I'm Password.";

[StringStyle(StringStyleType.Tag)]
public string tagString = "Untagged";

[StringStyle("Speed/Low", "Speed/High")]
public string categoryString = "Speed/High";
```

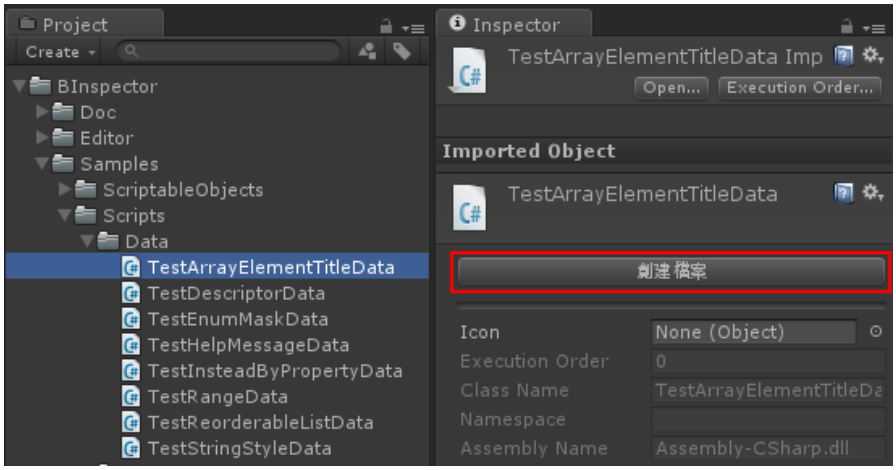


圖表 19. string 原始顯示方式

圖表 20. string 依照模式不同改善顯示方式

Scriptable Object Create Button

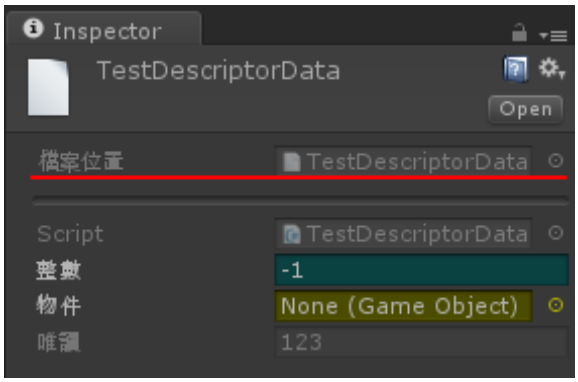
目的: 直接於程式碼檔案的 Inspector 上創建該 ScriptableObject



圖表 21. 不需額外寫函式來新增 ScriptableObject

Scriptable Object File Location Inspector

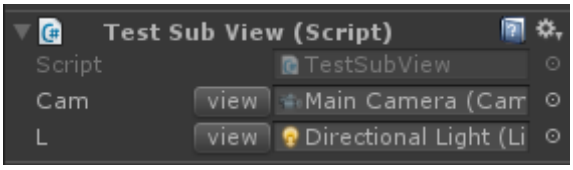
目的: 在 ScriptableObject 上顯示檔案位置，以便快速回到此檔案路徑



圖表 22. 顯示檔案位置。點選以便回到該路徑

Sub View

目的: 在拖曳欄位新增預覽按鈕，預覽該物件參數設定

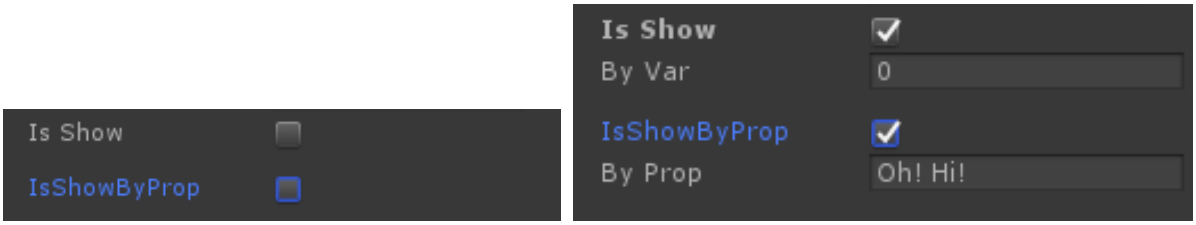


圖表 23. 新增預覽按鈕

圖表 24. 開啟預覽視窗

Conditional Hide

目的: 依照條件決定是否顯示該變數



圖表 25. 利用其餘變數條件來顯示該變數

使用方法:

```
// Parameters:
//   VariableName: 要判斷的布林 Field or Property or Method 名稱
//   IsInverse: 是否倒置觸發條件 (False 才觸發)

public bool isShow = false;

[BInspector.ConditionalHide("isShow", _isInverse: false)]
public int byVar = 0;
```