注意:在閱讀本文檔時，若出現 Sample: **.cs 便代表你能夠在 Sample 資料夾中找到相應的程式 API 範例，在此以解釋概念為主，請在找到需要的相關物件後前往查看。**

目錄

UIElementExtension 主要提供的功能為：

1. Runtime UITK Style Operation
2. Simple Localization
3. Runtime Drawer
4. Runtime Window
5. Useful VisualElement

Runtime UITK Style Operation

UITK 的外觀操作必須由 UIBuilder 或者在 Runtime 進行設定，但 Runtime 的 API 非常不友好，我們提供了一系列的函式來對此進行操作，包括：設定常用合集、複製、應用、線性差值等。

Simple Localization

你可以輕易的編輯本地化文本或圖片，並使用內置的 ISLocalizeText、ISLocalizeTextElement 等功能來進行 UIElement 的套用。

Runtime Drawer

一個能夠在 Runtime 使用物件繪製系統，能夠輕易地對物件進行實時更改、Debug，也能夠用來製作 In Game Editor，並且可以輕易的對你的自定義型別擴展

Runtime Window

一個能夠在 Runtime 使用的視窗系統，使用上類似於 Editor Window，可以輕易地製作各種彈出介面、編輯介面等。

Useful VisualElement

一些常用的控件合集，讓你不用再自行設計常見 Layout，如 SplitView、GridView、SearchView 等。

UIElementExtension 的功能並不是完全免費的，你可以通過下圖來了解

免費 付費 部分免費

NaiveAPI Core

- NaiveAPI Cache
- Type Reader

IS Group

- ISStyle
- ISTheme
- ISTransition
- ISLocalization

Runtime Window

- Runtime Window
- Runtime Scene Hierarchy
- Runtime Inspector

Runtime Drawer

- Bool Drawer
- Number Drawer
- Default Drawer
- String Drawer
- Type Drawer
- Enum Drawer
- GameObject Drawer
- UnityObject Drawer
- ISLocalization Drawer
- Array Drawer
- Color Drawer

Visual Element

- IS Element
- Popup Element
- Tooltip Element
- Search View
- Grid View

Runtime UITK Style Operation

UITK 的外觀操作必須由 UIBuilder 或者在 Runtime 進行設定，但 Runtime 的 API 非常不友好，我們提供了一系列的函式來對此進行操作，包括：設定常用合集、複製、應用、線性差值等。

如何在 Runtime 快速的操作 IStyle

- ISLength
- ISStyleComponent
- ISStyle

如何建立簡易的 VisualElement 動畫

- ISTransition

Sample: _01_ISLength.cs

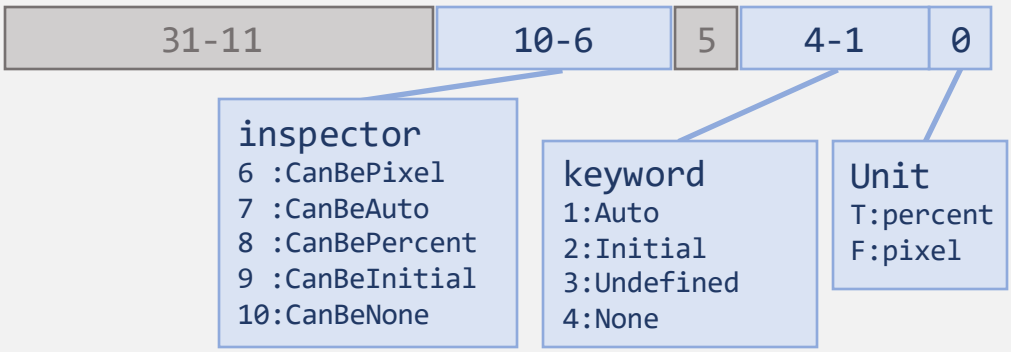
ISLength

在原生的 UITK 系統中，長度可以用三種型別來表達：float/Length/StyleLength，在本套件中統一為 ISLength，且可以對其他三種型別進行隱含轉換。同時包含了在 Inspector 上 Layout 的參數，並將其壓縮成 int32 flag 儲存。

***該內容並不是在使用時必須知道的**

ModeFlag：

ModeFlag 用以儲存當前的 keyword, unit, inspector setting 儲存的格式如下



(keyword, unit)的運算如下：

```
if keyword is Undefined:
      result is unit
else: result is first True in keyword
```

Inspector 可選模式：

*僅限定Editor時的數值設定，在 Runtime 你依然可以強制改變

Sample: _02_ISStyleComponent.cs

ISStyleComponent

這是一系列在 Runtime 對 IStyle 進行操作的 API，所以相關的 Class 皆繼承於 ISStyleComopnent。

基本介紹：

ISStyleComponent 對各種 Style 子類進行分開儲存，例如 ISSize 包含了長寬、最大、最小值，ISBorder 則包含了邊框設定，ISStyle 則包含了所有子類，下列是所有的 ISStyleComponent。

- ISStyle (include all below)
  - ISDisplay
  - ISPosition
  - ISFlex
  - ISAlign
  - ISSize
  - ISMargin
  - ISPadding
  - ISText
  - ISBackground
  - ISBorder
  - ISRadius
  - ISTransform

主要功能：

- 在 Runtime 時更輕易的編輯 IStyle 相關資料
- 複製已存在 VisualElement 的 Style
- 將編輯好的 Style 應用於 VisualElement
- 進行插值運算，實現簡易動畫

重要參數：SetUnsetFlag

SetUnsetFlag 用於決定哪些 Properties 是啟用的，並使用 int32 來進行儲存，例如 ISSize 包含6種參數，但通常只會需要使用高度和寬度，因此此時便會設定 SetUnsetFlag，讓進行相關操作時無視其他的參數。

Sample: _03_ISStyle.cs

ISStyle

ISStyle 是唯一一個操作特性和其他 ISStyleComponent 不同的型別，其他型別所儲存的都是各自對應的子類，而 ISStyle 儲存的是其他型別本身，因此雖然整體概念上以及 API 應用上是相同的，但若是你需要對相關物件做批量深度操作，必須注意這一點。

重要概念：

其他一般的 ISStyleComponent 不管 SetUnsetFlag 設定如何，其對應的數據大多也依然存在(應大多是 ValueType)，但對於 ISStyle，當一個數值被 Unset 時，其數據會變為 null，你必須手動幫其建構，或者重新 Set 其對應的 Flag。

Sample: _04_ISTransition.cs

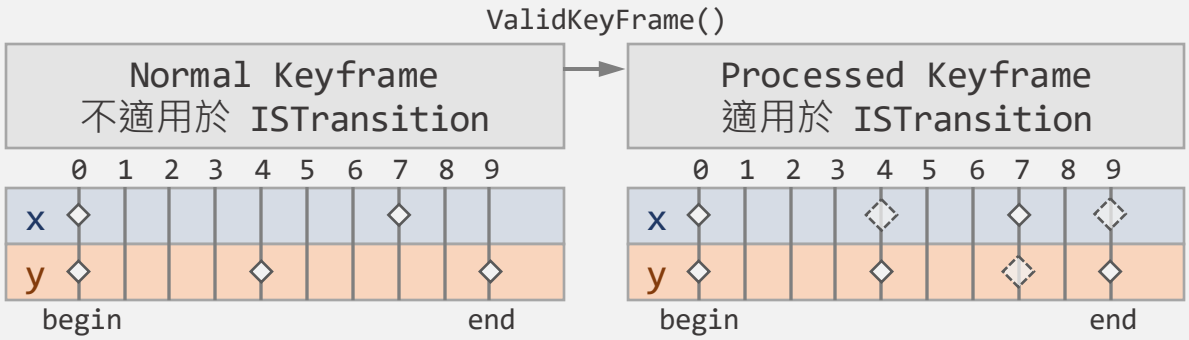
ISTransition

一個簡易的 VisualElement 動畫系統，目前僅支援在 Runtime 時進行創建、執行，目前並沒有提供 InEditor 預編輯功能的計畫，但可能會在未來進行更新。

重要概念：ValidKeyframe()

不同於一般的 Unity Animation，ISTransition 使用非常簡易的方式儲存資料，而在不同 Keyframe 之間，必須要使兩者的 Flag 一致才能有正常的撥放行為(當然你可以自己處理所有部分)，所以在設定完成後，你需要調用 ValidKeyframe() 將其中的數據正規化，也因此，若是你需要動態的在撥放 Transition 時進行 Style 更改，你必須確保相關的變數都被處理。

下圖可以更好的理解其原因：



Sample: _05_SimpleLocalization.cs

Simple Localization

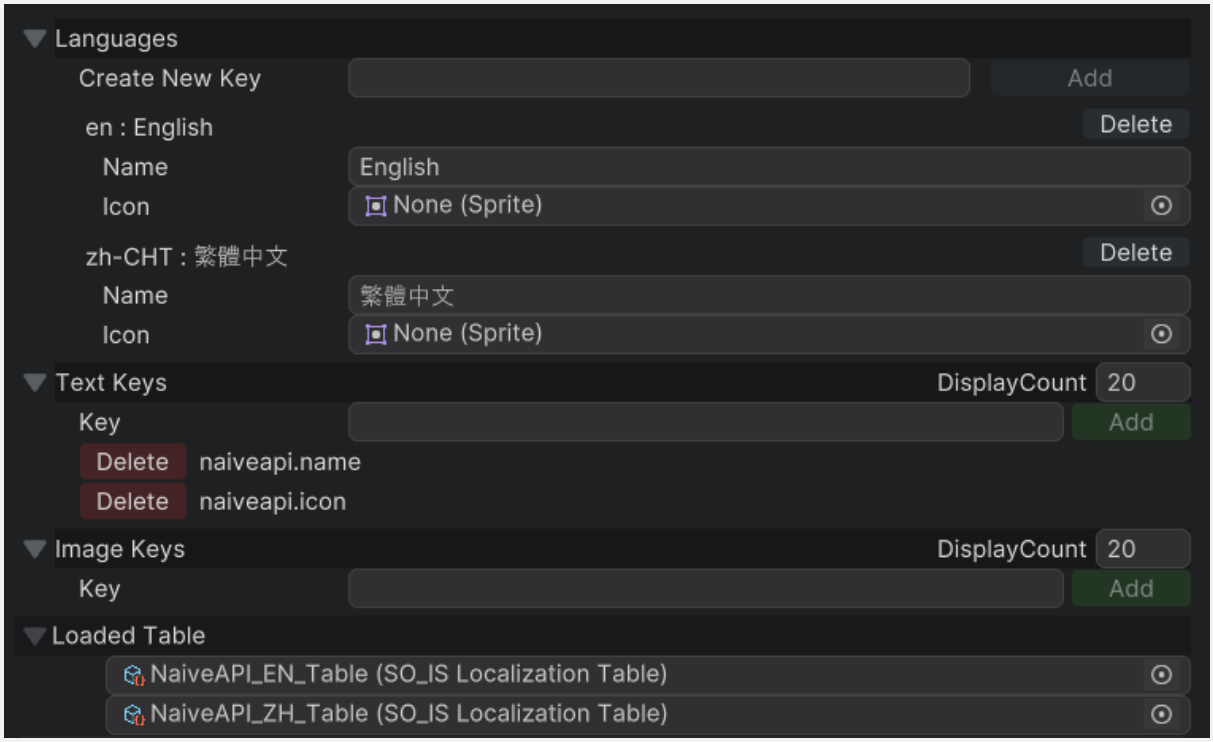
你可以輕易的編輯本地化文本或圖片，並使用內置的 ISLocalizeText、ISLocalizeTextElement 等功能來進行 UIElement 的套用。

各類別基本介紹：

ISLocalization: 提供主要本地化功能
ISLocalizeText: 可作為一般 string 或者 bind key 使用
ISLocalizeTextElement: 相應的 VisualElement

SO_ISLocalizationTable: 儲存各語言文本的實際數值
ISLocalizationKeyProvider: 通過程式提供 key 值的方式

在調用 API 取得本地化文本之前，你需要進行相應的設置
詳見下頁。



Sample: _05_SimpleLocalization.cs

Main Settings

Tools / Naive API / Settings -> IS Localization

設定語言種類，預設提供
en : English
zh-CHT : 繁體中文

設定 Key 值，稍後你能夠通過 Key 值取得文本
*你也可以使用 ISLocalizationKeyProvider 來藉由程式添加 Key
詳細用法見 **_05_ISLocalization.cs**

檢視當前成功加載到的 Key-Value Table

Sample: _05_SimpleLocalization.cs

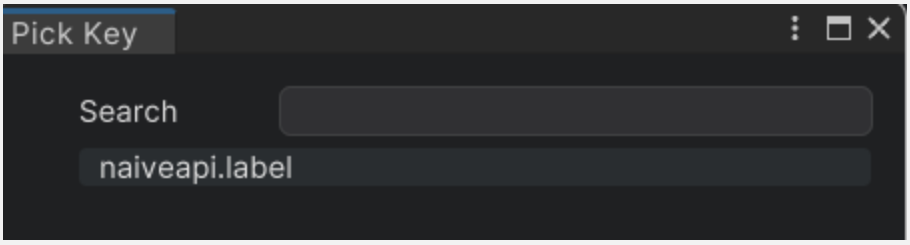
Create ScriptableObject

Create / Naive API / Localization Table

當前 Table 對應的語言	Language
當前 Table 對應的字體	Font Asset None (Font Asset)
下方顯示設定	Mode Text Display Count 50
添加或尋找 key 值	Search Key name Pick not Defined Add
編輯 key 值 所對應的文本	naiveapi.name Delete naiveapi.icon Delete

Pick not Defined

你可以通過 Pick not Defined 來直接尋找在當前語言中(不僅限當前的Table)，未提供文本的預定義 key 值。



*在添加文本時，並無限定只能添加預定義 key 值的文本。
完成設定後如何在程式中調用請見：**_05_SimpleLocalization.cs**

Runtime Drawer

一個能夠在 Runtime 使用物件繪製系統，能夠輕易地對物件進行實時更改、Debug，也能夠用來製作 In Game Editor，並且可以輕易的對你的自定義型別擴展

目錄

客製化

Runtime Drawer

一個能夠在 Runtime 使用物件繪製系統，能夠輕易地對物件進行實時更改、Debug，也能夠用來製作 In Game Editor，並且可以輕易的對你的自定義型別擴展

與原生 UIElement Field 的差別

Unity 所提供的原生控件僅限於常見的 ValueType，並且我們難以輕易的對其版面配置進行設定，在 Runtime 設定 Style 時也非常困難，而 RuntimeDrawer 可以對任何型別進行擴展，並使用 ISTheme 作為 Style 來源，能夠輕易設定。更多比較見下表：

控件對比	Runtime Drawer	Unity Field
Unity 版本支援	2021+	2022+
自定義排版	容易	困難
擴展難易度	容易	困難
Runtime 支援	較好	較差
型別自動建置	可以	不行
動態 Layout	可以	不行
運行效率	較差	較好

Sample: _06_RuntimeDrawer.cs

使用方式

RuntimeDrawer 有數種常用方式：
1. 作為單純 Field 使用
2. 作為物件 Editor 使用

基本介紹：

一個完善定義的 RuntimeDrawer 具備以下重要功能：

1. value
   當前繪製的物件

2. event OnValueChanged
   該事件會在 value 改變時呼叫

3. event OnMemberValueChanged
   該事件會在成員變量改變時呼叫
   (僅在 value 為 ReferenceType 時觸發)

你可以通過設定 value 與註冊事件，即可輕易地完成 Editor 或 Inspector 的建置。

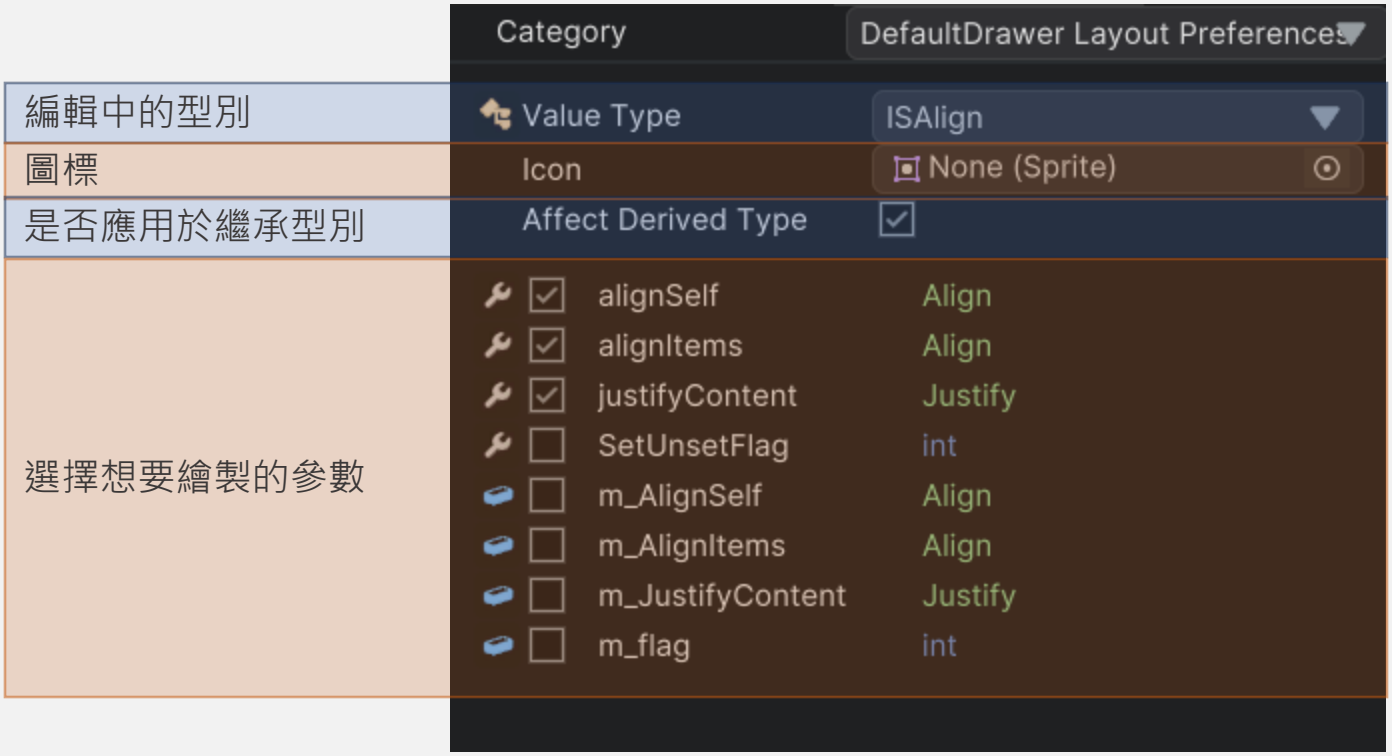
Default Drawer

你可以使用 RuntimeDrawer.Create(object) 來對任意型別物件進行 Drawer 建構，若是找不到專屬的 Drawer，便會套用 DefaultDrawer，預設繪製與 Unity Inspector 一致，你可以通過自定義 Drawer (見後續客製化內容)，或是直接通過改變預設設定來決定要顯示那些參數(見下頁)。

Sample: _06_RuntimeDrawer.cs

Default Drawer Settings

Tools / Naive API / Settings
-> Default Drawer Layout Preference



Sample: _07_CustomRuntimeDrawer.cs

客製化：

要客製化你的 RuntimeDrawer，有下列基本函式需要實作
- CreateGUI()
- RepaintDrawer()

以下是可選項目
- Construct with Attribute
- Dynamic Layout
- Indent Layout

詳細的客製化方式，見後續頁面。

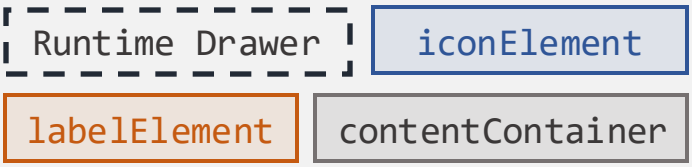
Sample: _07_CustomRuntimeDrawer.cs

Layout mode

RuntimeDrawer 有兩種 Layout 模式，Inline/Expand，通過調用各自的函式能夠改變當前的排版：LayoutInline() / LayoutExpand() 同時這也是可以 override 的部分。

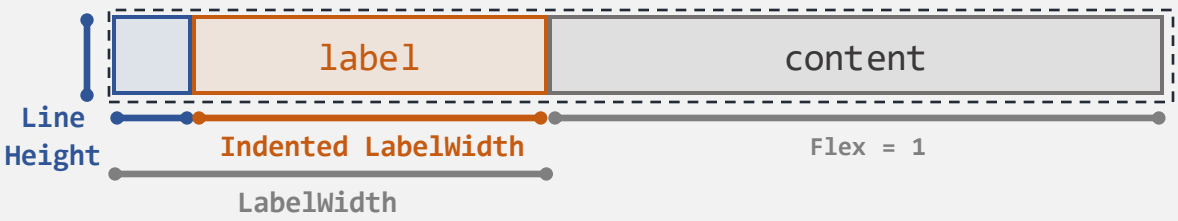
Default Layout

RuntimeDrawer 將會預建置數個控建，你能夠直接從成員變數調用到他們，以下是他們的預設 Layout，你所 Add() 的物件將會被加入到 contentContainer。

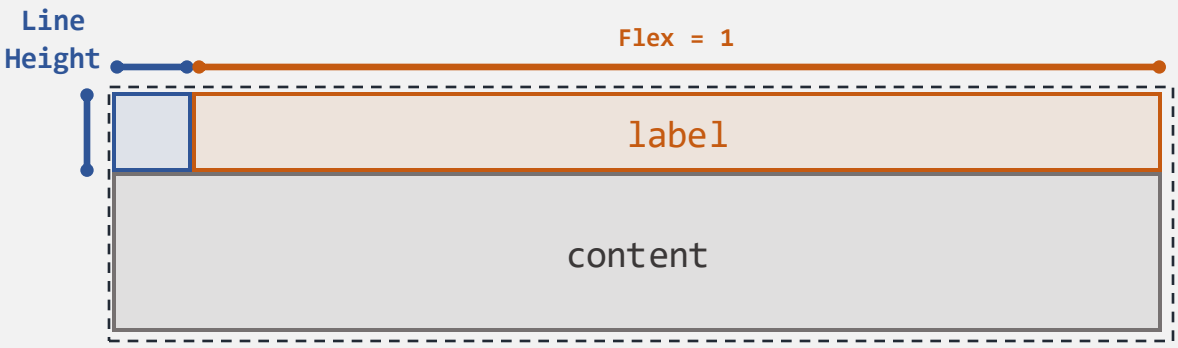


*此處使用到的參數由 ISTheme 設定

Layout Inline



Layout Expand



*對於部分內建 Drawer，Inline/Expand 不只會影響排版，可能還影響其他內容，例如 UnityObjectDrawer 在 Inline 模式下繪製的是 Reference， Expand 模式下是繪製其成員變量。

Sample: _07_CustomRuntimeDrawer.cs

1. 選擇繼承型別

目前有4種預設型別可供繼承，可以照需求選擇
- 完整實作：RuntimeDrawer<T>
- 完整實作：FoldoutDrawer<T> : RuntimeDrawer<T>
- 快速實作：StandardDrawer<T> : FoldoutDrawer<T>
- 快速實作：UnityObjectDrawer<T> : StandardDrawer<T>

*對於任何 Unity.Object，都建議使用 UnityObjectDrawer<T>

2. 實作功能函式

```
bool override DynamicLayout { get; }
```
*預設為 false

建置編輯GUI：CreateGUI()
刷新GUI顯示：RepaintDrawer()

以下是其被呼叫的時機
CreateGUI:
DynamicLayout is True : value 改變時呼叫
False : 在建構時呼叫
RepaintDrawer:
- 外部直接調用時
- 調用完 CreateGUI 且 value 不為 null 時

3. (*可選) 註冊 Drawer

通過宣告 CustomRuntimeDrawerAttribute 來註冊你的 Drawer
當然不註冊也能夠使用，但僅能夠自行建構，註冊後能通過
RuntimeDrawer.Create() 來進行動態建購。

CustomRuntimeDrawerAttribute Properties
- targetType 該 Drawer 所渲染的 value 型別
- Priority 建構的優先級
- DrawDerivedType 是否繪製繼承 targetType 的子型別
- DrawAssignableType 是否繪製能轉型成 targetType 的子型別
- RequiredAttribute 需要指定 Attribute 進行建構，見下頁

Sample: _07_CustomRuntimeDrawer.cs

4. (*若步驟 3 中的 RequiredAttribute != null)

若是有額外資訊才能進行 Drawer 建構，就必須註冊且設定 RequiredAttribute 的型別，例如可用於 int/float 的 RangeAttribute，最後 override ReciveAttribute() 函式來接收資訊。相關範例可參考 FloatRangeDrawer 。

*只有使用 RuntimeDrawer.Create() 時才會生效。
直接使用建構子建構並不會觸發該效果

Sample: _08_RuntimeDrawerDecorator.cs

DrawerDecorator

若是有與 Drawer 無直接相關的修飾性 Attribute，例如 Header/Tooltip，便可宣告 Decorator 進行修飾。

*只有使用 RuntimeDrawer.Create() 時才會生效。
直接使用建構子建構並不會觸發該效果

Runtime Window

一個能夠在 Runtime 使用的視窗系統，使用上類似於 Editor Window，可以輕易地製作各種彈出介面、編輯介面等。

基本介紹

Runtime Window 是 Unity Editor Window 在 Runtime 的簡易實現，若是你有相關的使用經驗，那基本是一致的。

*關於 RuntimeWindow 的使用，請直接前往程式 API 以及範例場景觀看。

內建功能 Window

本套件提供了一些預建置好的 RuntimeWindow 可供使用：

- **RuntimeInspector**
  利用 RuntimeDrawer，建立可在 Runtime 即時觀看且調適的 Inspector 視窗。

- **RuntimeSceneHierarchy**
  能夠渲染當前場景物建結構的視窗，且可與 RuntimeInspector 綁定來選擇顯示對象。