Rev 2a

2025-01-11

UI Toolkit

Danai Jedsadathitikul

Jarut Busarathid

DcG.IT@PBRU

การใช้งาน UI ใน Unity เป็นกระบวนการสำคัญสำหรับการสร้างอินเทอร์เฟซผู้ใช้ที่มีประสิทธิภาพในเกมหรือแอปพลิเคชัน

Unity มีระบบ UI ที่ทรงพลังซึ่งรองรับการสร้าง
องค์ประกอบ UI เช่น

- ปุ่ม (Buttons)
- ข้อความ (Text)
- ภาพ (Images)
- และอื่น ๆ

อย่างง่ายดายผ่าน Canvas System และ Event System

UI Toolkit

- UI Toolkit เป็นระบบ UI ที่พัฒนาต่อเนื่องใน Unity
- เหมาะสำหรับการสร้าง UI ที่ปรับขนาดได้ง่าย เช่น เมนู การตั้งค่า และ UI สำหรับ Editor

- เป็นระบบ UI ใหม่ที่ออกแบบมาเพื่อช่วยให้การสร้าง UI เป็นเรื่องง่ายทั้งสำหรับ Editor UI และ Runtime UI (UI ที่แสดงผลขณะเล่นเกม)
- ใช้แนวคิดที่คล้ายกับการพัฒนาเว็บ เช่น UXML (คล้าย HTML) และ USS (คล้าย CSS) ในการกำหนดโครงสร้างและการออกแบบ

คุณสมบัติเด่น

- ใช้ USS (Unity Style Sheets) ที่คล้ายกับ CSS ในการจัดรูปแบบ
- รองรับ UXML สำหรับการกำหนดเลย์เอาต์ (คล้าย HTML)
- ผสานเข้ากับระบบ Editor UI และ Runtime UI
- ประสิทธิภาพสูงเมื่อเทียบกับ Unity UI แบบเก่า

UXML (Unity XML)

- ใช้สำหรับสร้างโครงสร้างของ UI (คล้าย HTML)
- แต่ละองค์ประกอบใน UI เช่น ปุ่ม หรือข้อความจะถูกกำหนดในไฟล์ .uxml

USS (Unity Style Sheets)

- ใช้สำหรับกำหนดรูปแบบ (Styling) ของ UI (คล้าย CSS)
- เช่น การตั้งค่าแบบอักษร สี ขนาด ฯลฯ

UI Builder

- เป็นเครื่องมือแบบ Visual ที่ช่วยให้คุณสร้าง UI ได้ง่ายโดยไม่ต้องเขียนโค้ด
- สามารถดูผลลัพธ์ UI ได้แบบเรียลไทม์

UIDocument

- ใช้แสดงผล UI ใน Runtime โดยเชื่อมโยง .uxml และ .uss เข้ากับ Scene

ข้อดีของ UI Toolkit

- ปรับขนาด UI ได้ง่าย: ด้วย Flexbox คล้าย CSS
- แยกส่วนโครงสร้างและสไตล์: ช่วยให้จัดการง่ายขึ้น
- ประสิทธิภาพสูง: ออกแบบมาให้เหมาะสำหรับ UI ขนาดใหญ่

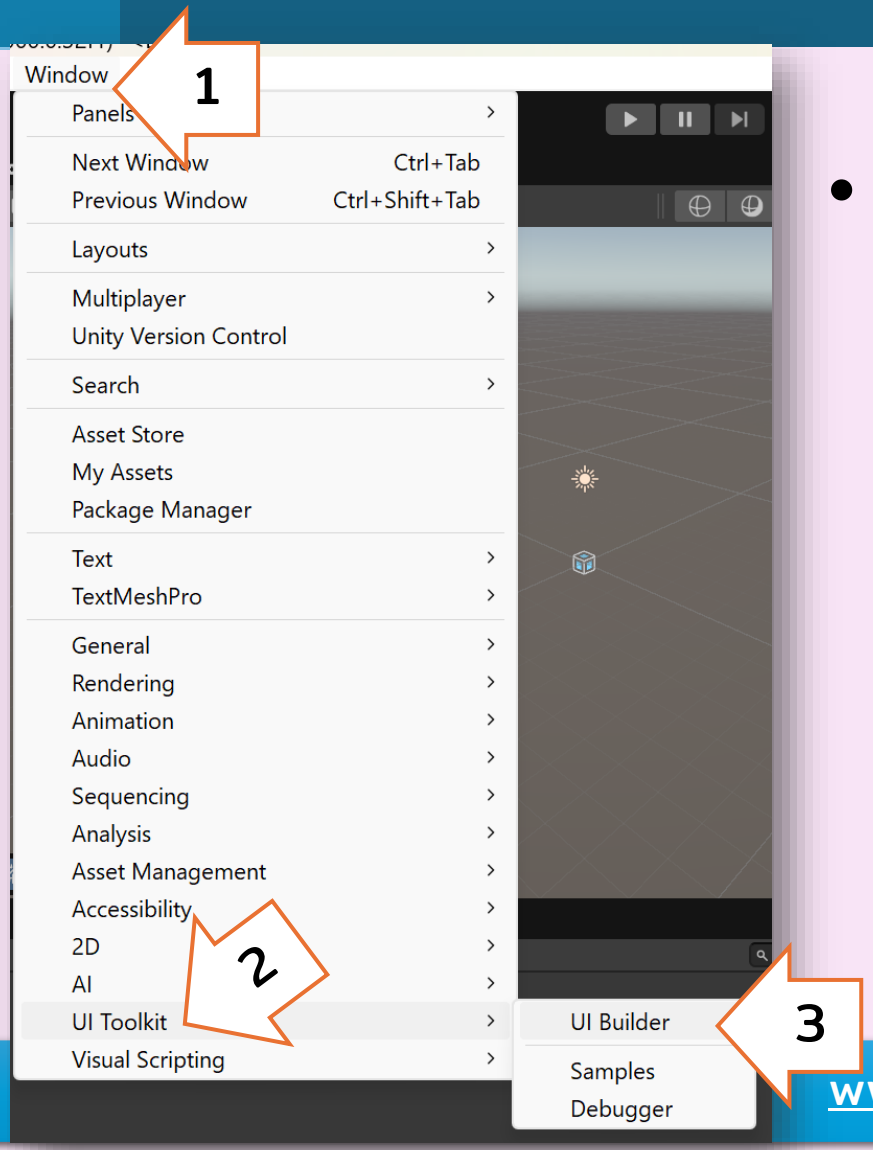
การนำ UI Toolkit ไปใช้

- ใช้สำหรับ Runtime UI เช่น เมนูหลัก, HUD (Head-Up Display)
- ใช้สร้าง Editor Extensions เพื่อปรับแต่ง Editor ของ Unity
- ใช้งานร่วมกับ Unity UI (uGUI) ได้หากจำเป็น

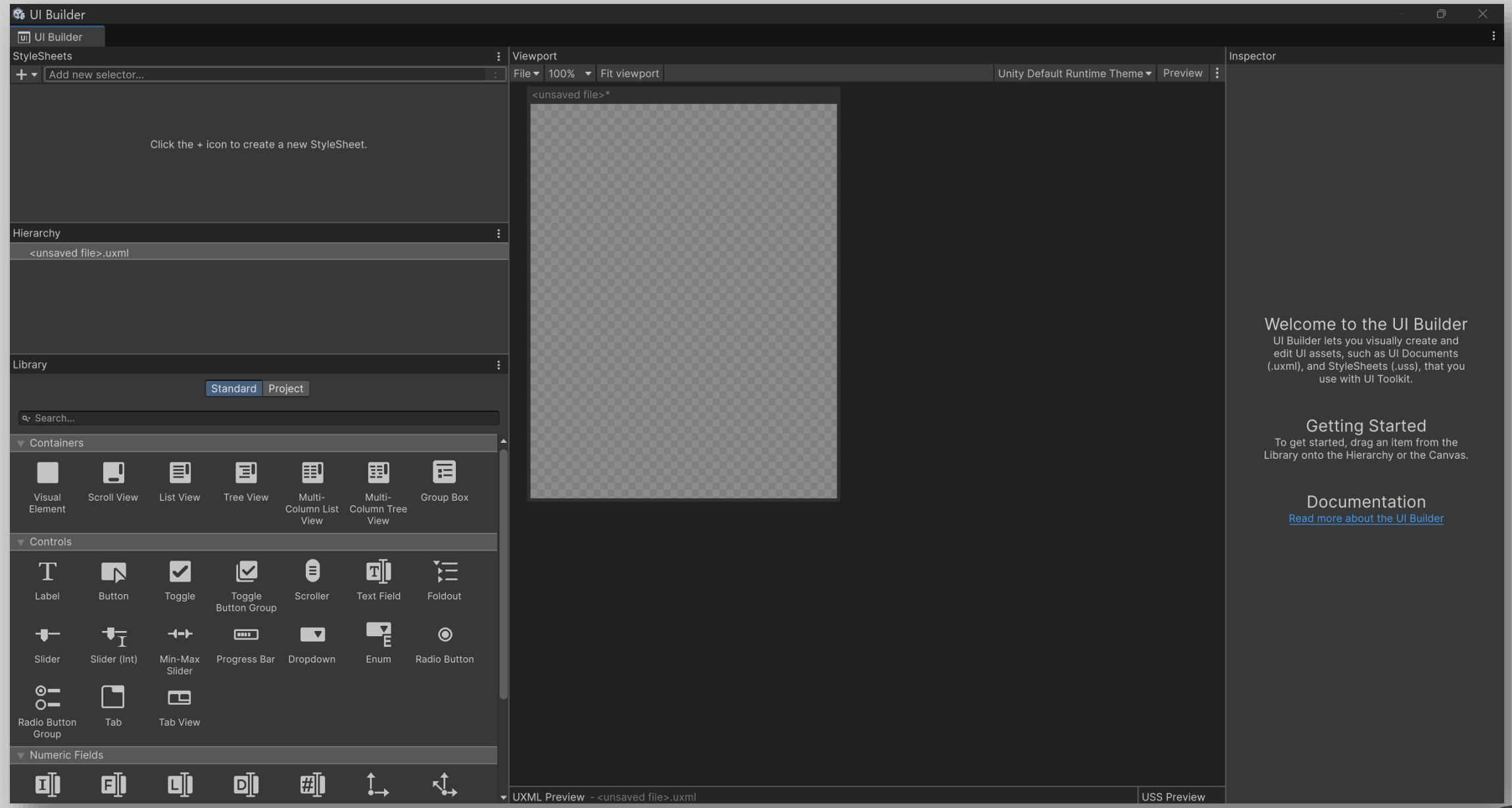
วิธีเริ่มต้นใช้งาน

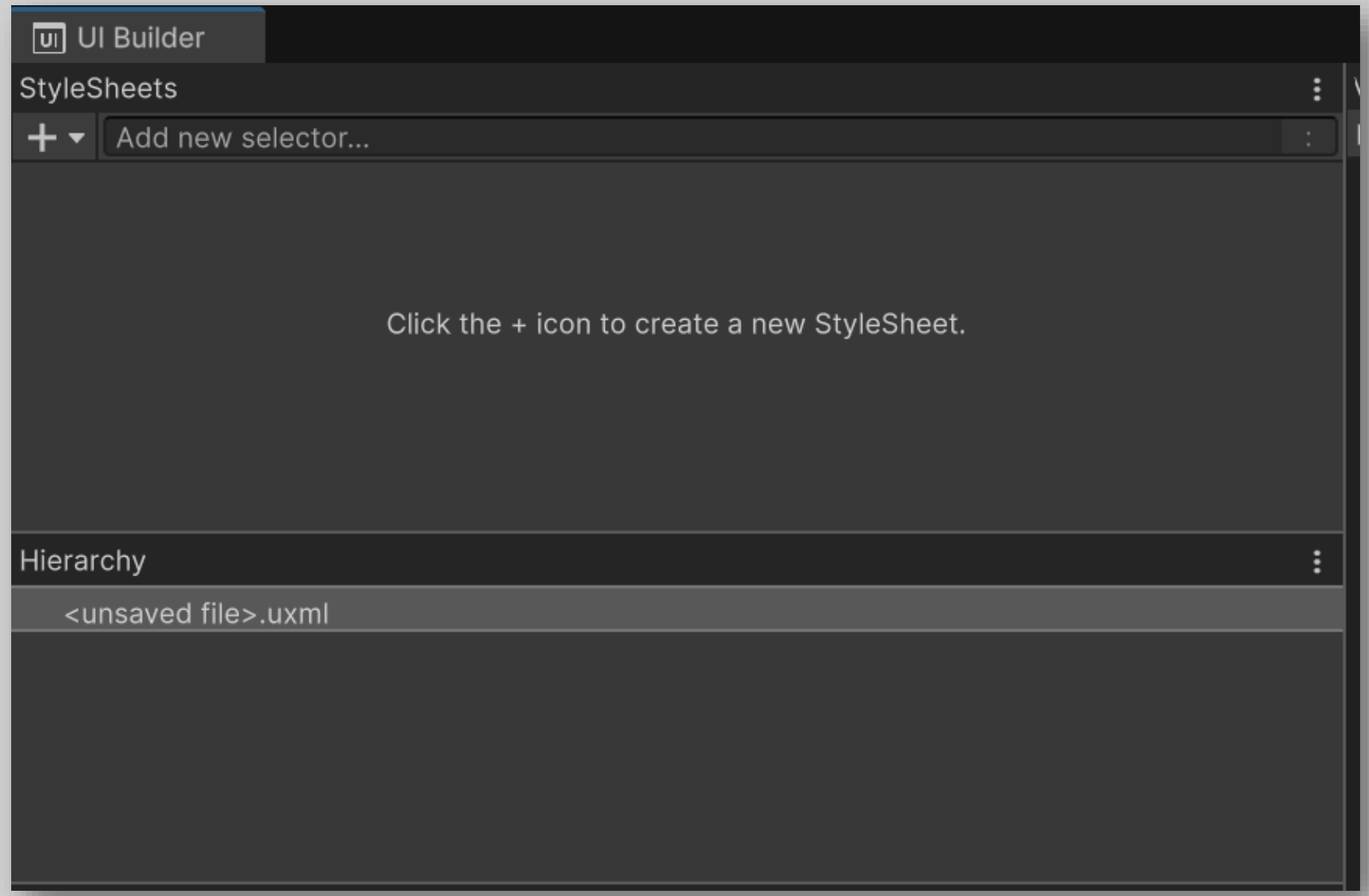
www.gamedevpbru.com

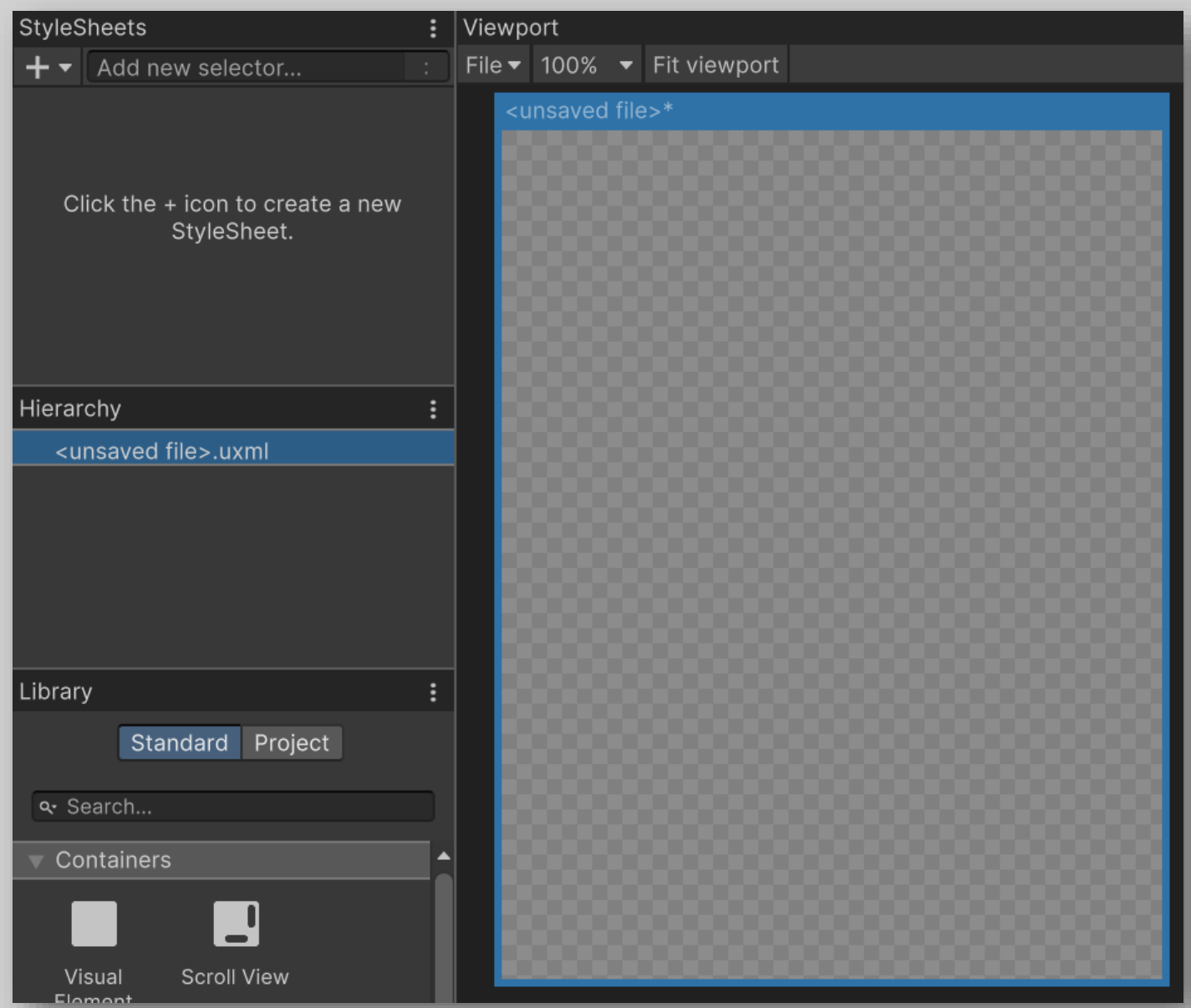
1 เปิดใช้งาน UI Toolkit

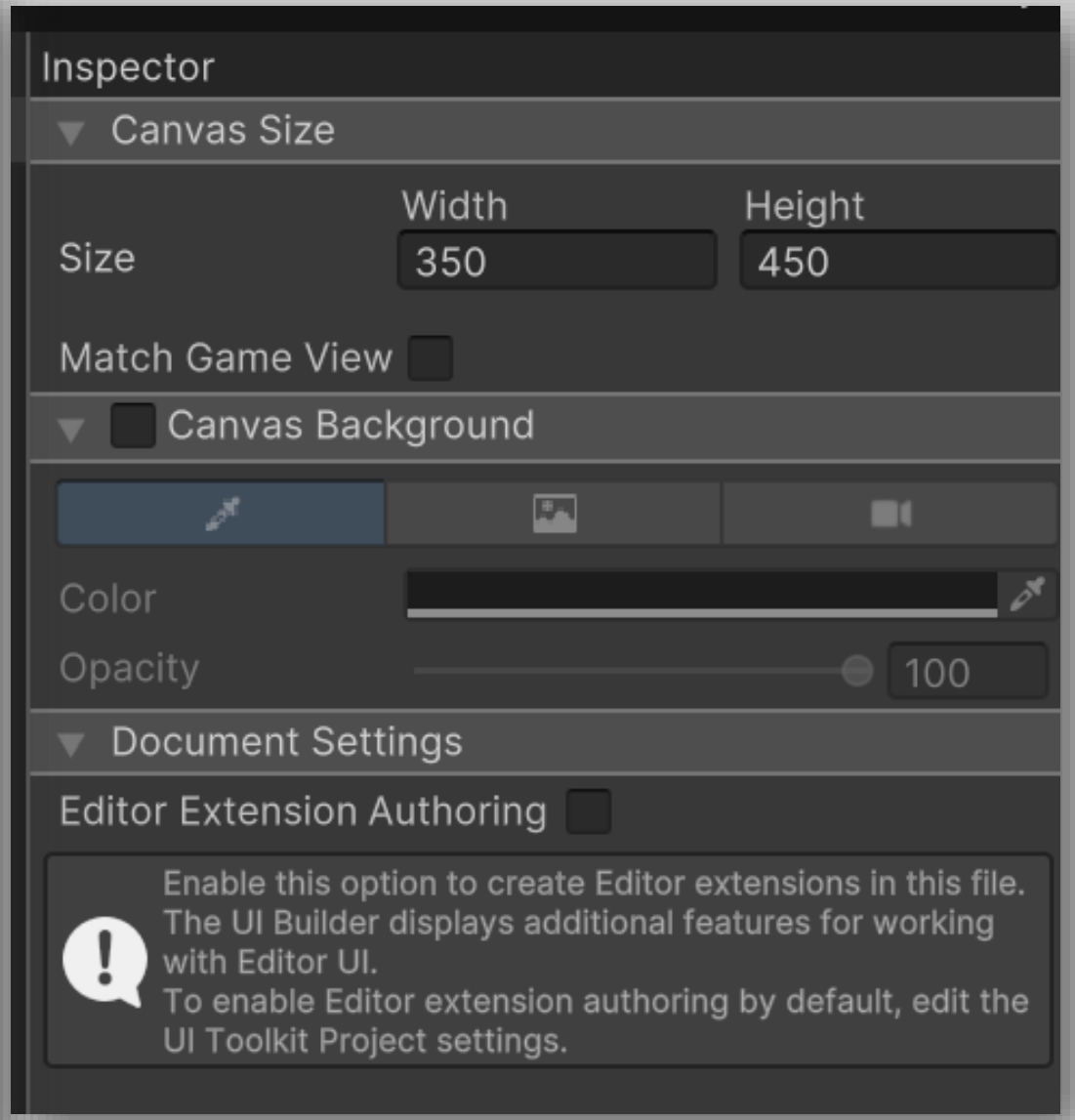


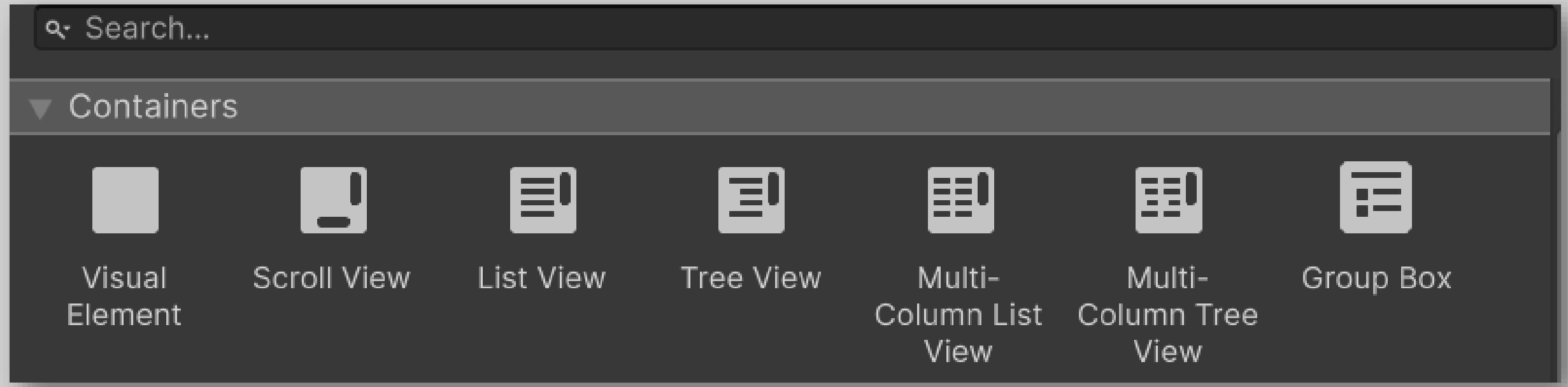
- ไปที่ Window > UI Toolkit > UI Builder เพื่อสร้างและออกแบบ UI

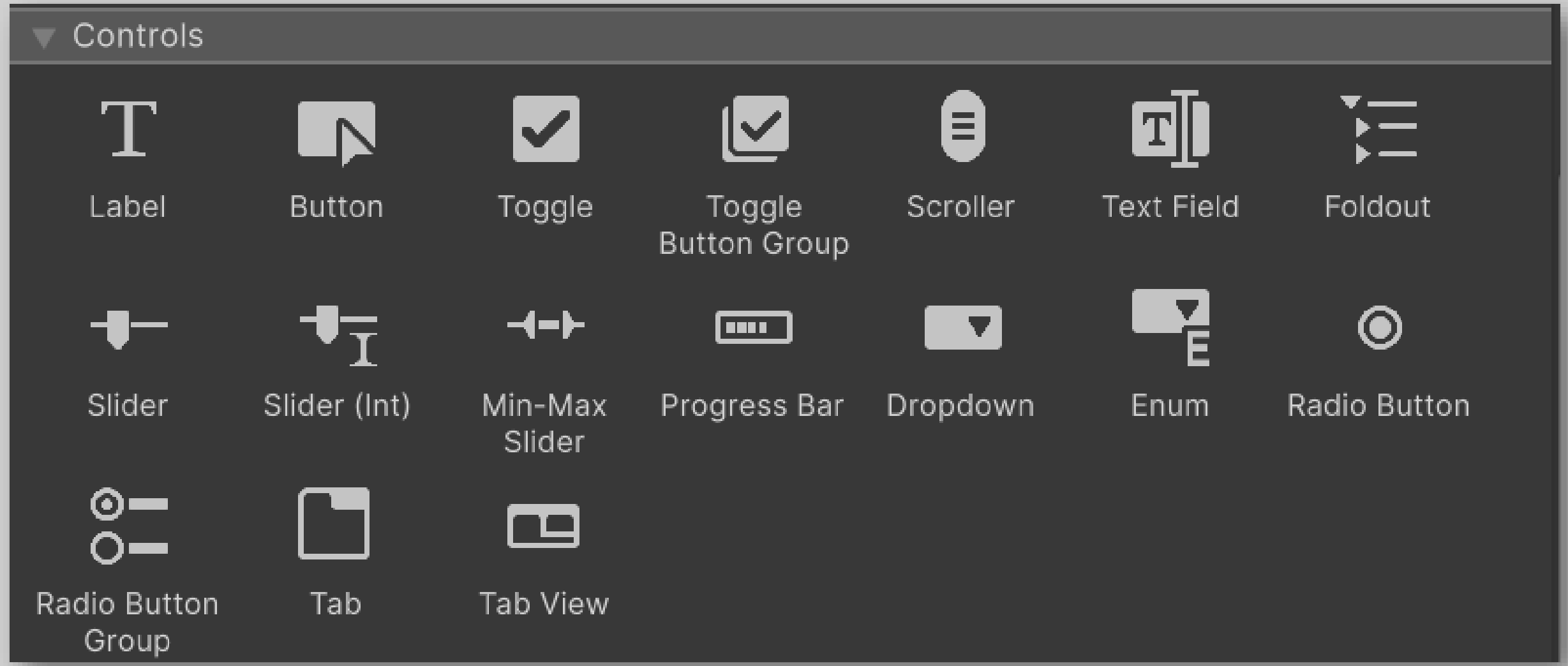

UI Builder
UI Builder
StyleSheets
Add new selector...
Click the + icon to create a new StyleSheet.
Hierarchy
<unsaved file>.uxml
Library
Standard
Project
Search...
Containers
Visual Element
Scroll View
List View
Tree View
Multi-Column List View
Multi-Column Tree View
Group Box
Controls
Label
Button
Toggle
Toggle Button Group
Scroller
Text Field
Foldout
Slider
Slider (Int)
Min-Max Slider
Progress Bar
Dropdown
Enum
Radio Button
Radio Button Group
Tab
Tab View
Numeric Fields
Viewport
File
100%
Fit viewport
Unity Default Runtime Theme
Preview
<unsaved file>*
UXML Preview - <unsaved file>.uxml
USS Preview
Inspector
Welcome to the UI Builder
UI Builder lets you visually create and edit UI assets, such as UI Documents (.uxml), and StyleSheets (.uss), that you use with UI Toolkit.
Getting Started
To get started, drag an item from the Library onto the Hierarchy or the Canvas.
Documentation
Read more about the UI Builder

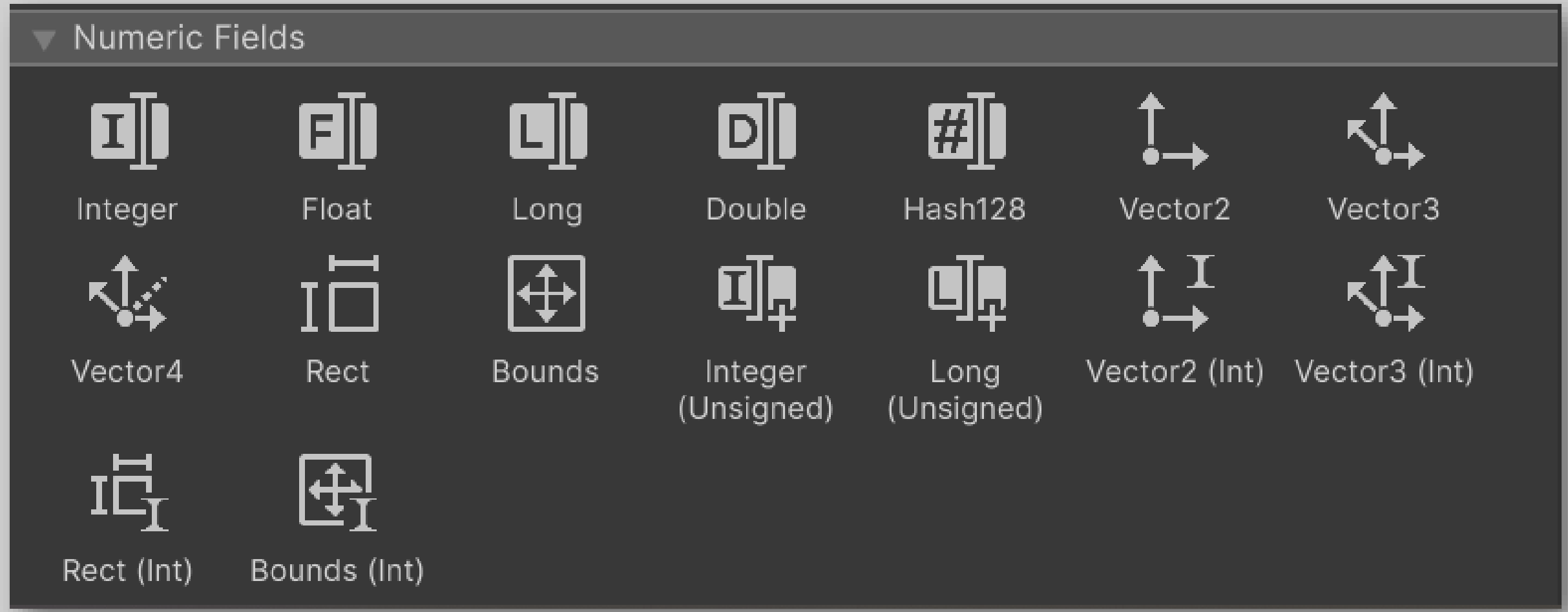

UI Builder
StyleSheets
Add new selector...
Click the + icon to create a new StyleSheet.
Hierarchy
<unsaved file>.uxml


StyleSheets
Add new selector...
Click the + icon to create a new StyleSheet.
Hierarchy
<unsaved file>.uxml
Library
Standard
Project
Search...
Containers
Visual Element
Scroll View
Viewport
File
100%
Fit viewport
<unsaved file>*


Inspector
Canvas Size
Width
Height
Size
350
450
Match Game View
Canvas Background
Color
Opacity
100
Document Settings
Editor Extension Authoring
Enable this option to create Editor extensions in this file. The UI Builder displays additional features for working with Editor UI.
To enable Editor extension authoring by default, edit the UI Toolkit Project settings.

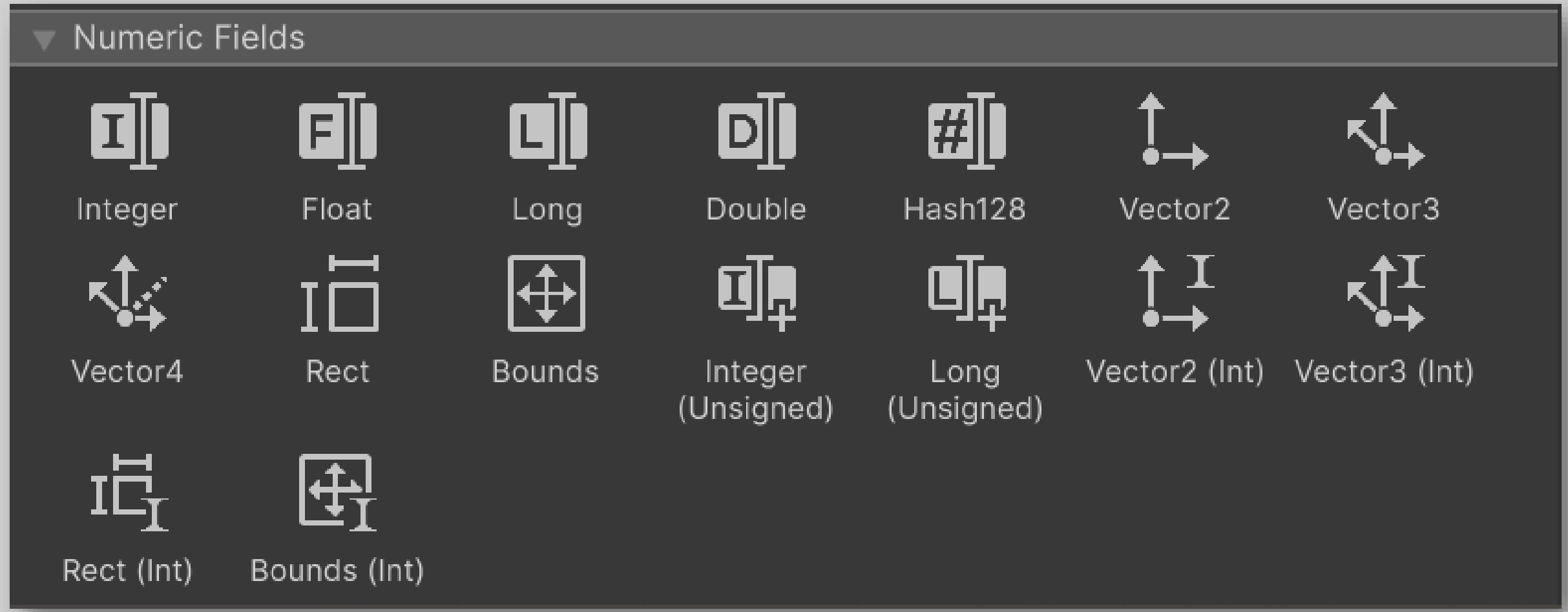

Search...
Containers
Visual Element
Scroll View
List View
Tree View
Multi-Column List View
Multi-Column Tree View
Group Box


Controls
Label
Button
Toggle
Toggle Button Group
Scroller
Text Field
Foldout
Slider
Slider (Int)
Min-Max Slider
Progress Bar
Dropdown
Enum
Radio Button
Radio Button Group
Tab
Tab View


Numeric Fields
Integer
Float
Long
Double
Hash128
Vector2
Vector3
Vector4
Rect
Bounds
Integer (Unsigned)
Long (Unsigned)
Vector2 (Int)
Vector3 (Int)
Rect (Int)
Bounds (Int)

2 สร้าง UI

- สร้างไฟล์ใหม่
  - คลิกปุ่ม New เพื่อเริ่มสร้างไฟล์ .uxml ใหม่
  - ตั้งชื่อไฟล์ เช่น MyLayout.uxml
- เพิ่ม UI Components
  - ลากและวางองค์ประกอบ (เช่น Label, Button, TextField) จาก Library ด้านซ้ายเข้าสู่พื้นที่ออกแบบ (Canvas)
  - สามารถปรับแต่งโครงสร้างใน Hierarchy ด้านขวา


Numeric Fields
Integer
Float
Long
Double
Hash128
Vector2
Vector3
Vector4
Rect
Bounds
Integer (Unsigned)
Long (Unsigned)
Vector2 (Int)
Vector3 (Int)
Rect (Int)
Bounds (Int)

- จัดสไตล์
  - คลิกที่องค์ประกอบที่ต้องการจัดสไตล์
  - ใน Inspector ด้านขวา เลือก Add StyleSheet เพื่อเพิ่มไฟล์ .uss ใหม่
  - ตั้งชื่อไฟล์ เช่น MyStyle.uss
- แก้ไข Style
  - ใน UI Builder สามารถแก้ไข Style ได้แบบ Visual หรือเปิดไฟล์ .uss เพื่อแก้ไขด้วยโค้ด
- บันทึกไฟล์
  - UI Builder จะสร้างไฟล์ .uxml และ .uss ไว้ในโฟลเดอร์ที่คุณกำหนด

3 แสดง UI ใน Scene

- สร้าง UIDocument ใน Scene แล้วเชื่อมโยงกับ .uxml

ตัวอย่าง

การสร้าง UI ด้วย UI Toolkit

www.gamedevpbru.com

1 การสร้าง UI Layout ด้วย UI Builder

- เปิด Unity และไปที่ Window > UI Toolkit > UI Builder
- ใน UI Builder:
  - สร้าง VisualElement ใหม่ และเพิ่มองค์ประกอบ UI เช่น Label หรือ Button
  - ตั้งค่าคุณสมบัติใน Inspector เช่น ข้อความ ขนาด หรือสี
- บันทึกเป็นไฟล์ .uxml และ .uss

2 ตัวอย่างไฟล์ UXML

```
<ui:UXML xmlns:ui="UnityEngine.UIElements">
      <ui:VisualElement class="root">
            <ui:Label text="Welcome to UI Toolkit!" class="title" />
            <ui:Button text="Click Me" class="button" />
      </ui:VisualElement>
</ui:UXML>
```

3 ตัวอย่างไฟล์ USS


```
.root {
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 100%;    height: 100%;
}
```

```
.title {
    font-size: 24px;
    color: white;
    margin-bottom: 10px;
}
.button {
    font-size: 18px;
    padding: 10px 20px;
    background-color: #3498db;
    color: white;
    border-radius: 5px;
}
```

4 เชื่อม UI กับ Scene

- สร้าง GameObject ใน Scene และเพิ่มคอมโพเนนต์ UIDocument
- ใน Inspector ของ UIDocument:
  - ตั้งค่าฟิลด์ Source Asset เป็นไฟล์ .uxml ที่สร้างไว้

การเขียน Script ควบคุม UI

www.gamedevpbru.com

- หากต้องการให้ UI ตอบสนองต่อการกระทำของผู้ใช้ (เช่น คลิกปุ่ม) สามารถใช้ C# Script ได้
- ตัวอย่าง Script สำหรับจัดการ Button

```
using UnityEngine;
using UnityEngine.UIElements;
public class UIController : MonoBehaviour{
    private Button myButton;
    void OnEnable()    {
        // ดึง UIDocument
        var uiDocument = GetComponent<UIDocument>();
        var root = uiDocument.rootVisualElement;
        // ค้นหา Button ที่มีชื่อว่า "button" จากไฟล์ .uxml
        myButton = root.Q<Button>("button");
        // เพิ่ม Event เมื่อกดปุ่ม
      myButton.clicked += OnButtonClick;
    }
```

```
    private void OnButtonClick()    {
        Debug.Log("Button clicked!");
    }
}
```

Unity UI (uGUI)

Unity UI แบบดั้งเดิมยังคงใช้งานได้และเหมาะสำหรับโปรเจกต์ที่มีอยู่หรือสำหรับงานที่ต้องการการปรับแต่งเฉพาะ

คุณสมบัติเด่น

- ใช้งานง่ายและคุ้นเคยสำหรับผู้ที่เคยใช้ Unity UI ก่อนหน้านี้
- รองรับการทำงานร่วมกับระบบ Input Manager หรือ Event System
- มีคอมโพเนนต์ที่พร้อมใช้งาน เช่น Button, Slider, Dropdown

1 Canvas

- Canvas เป็นรากฐานของ UI ใน Unity ทุกองค์ประกอบ UI จะต้องอยู่ภายใต้ Canvas
- มี 3 รูปแบบหลักของ Canvas
  - Screen Space - Overlay: องค์ประกอบ UI จะถูกวาดบนหน้าจอและไม่ขึ้นกับตำแหน่งของกล้อง
  - Screen Space - Camera: องค์ประกอบ UI จะถูกเรนเดอร์ตามกล้องที่ระบุ
  - World Space: องค์ประกอบ UI จะทำงานเหมือน GameObject ในโลก 3D

2 RectTransform

- RectTransform ใช้สำหรับกำหนดตำแหน่ง, ขนาด และสัดส่วนขององค์ประกอบ UI
- ใช้ Anchor, Pivot และ Position เพื่อควบคุมการจัดตำแหน่ง UI อย่างยืดหยุ่น

วิธีเริ่มต้นใช้งาน

www.gamedevpbru.com

1 สร้าง Canvas

- ไปที่ GameObject > UI > Canvas เพื่อเริ่มต้นสร้าง UI

2 เพิ่ม UI Elements

- เพิ่มคอมโพเนนต์ เช่น Text, Image, Button ลงใน Canvas

3 ปรับแต่ง

- ใช้ RectTransform เพื่อจัดการตำแหน่ง ขนาด และการยืดหยุ่นของ UI

4 เพิ่ม Script

- เขียน C# Script เพื่อควบคุมการทำงานของ UI เช่น การคลิกปุ่ม

การใช้งาน UI ให้มีประสิทธิภาพ

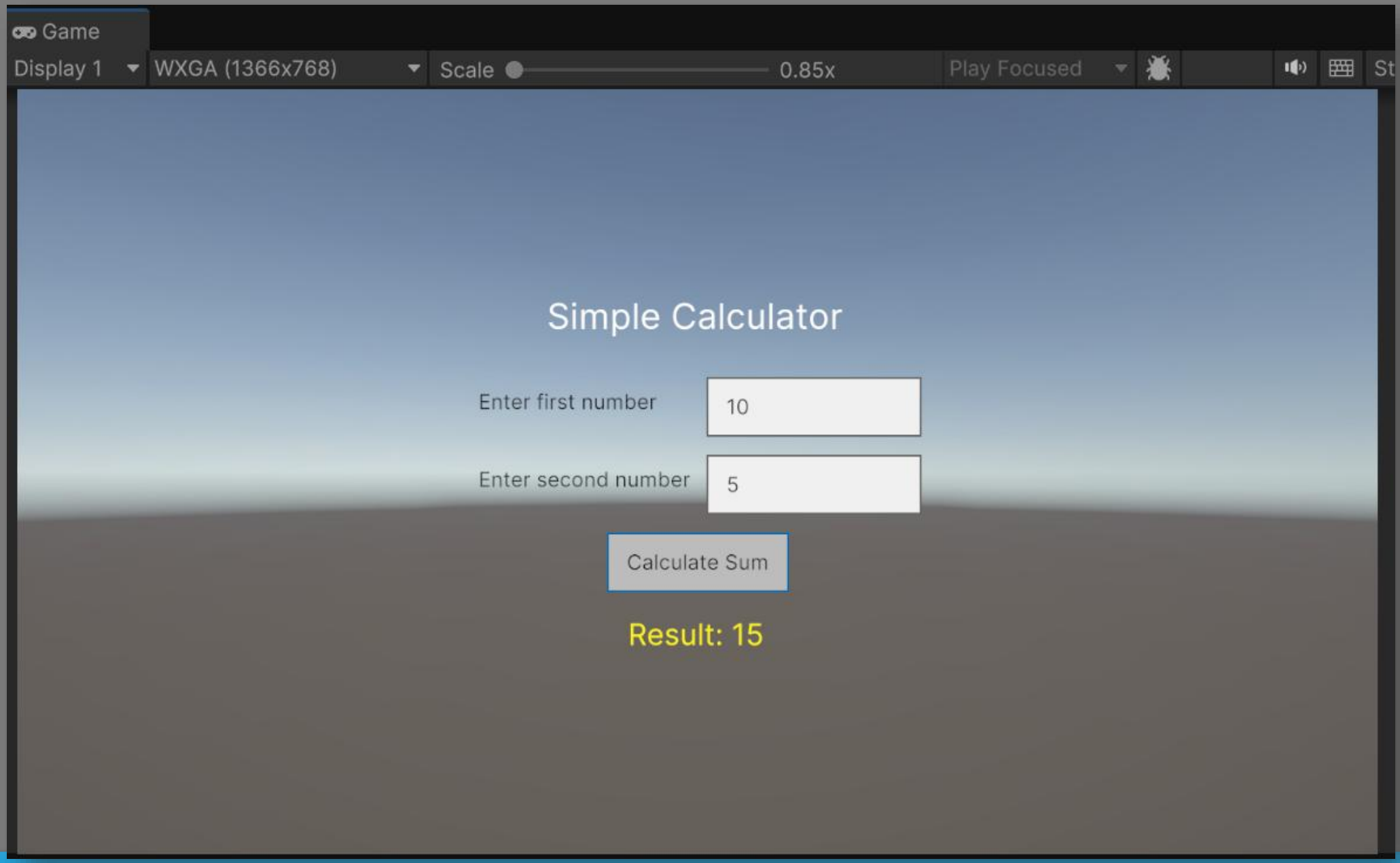
- ใช้ระบบ Layout: ใช้ Horizontal Layout Group และ Vertical Layout Group เพื่อจัดเรียงองค์ประกอบแบบอัตโนมัติ
- Anchor และ Pivot: ปรับ Anchor และ Pivot ของ UI Element ให้เหมาะสมเพื่อรองรับการปรับขนาดหน้าจอ
- จัดการ UI Scale: ใช้ Canvas Scaler เพื่อปรับ UI ให้เหมาะสมกับความละเอียดหน้าจอที่หลากหลาย

- ปรับปรุงประสิทธิภาพ
  - ลดจำนวน Canvas ที่ซ้อนกัน
  - ใช้ Sprite Atlas เพื่อลด Draw Calls

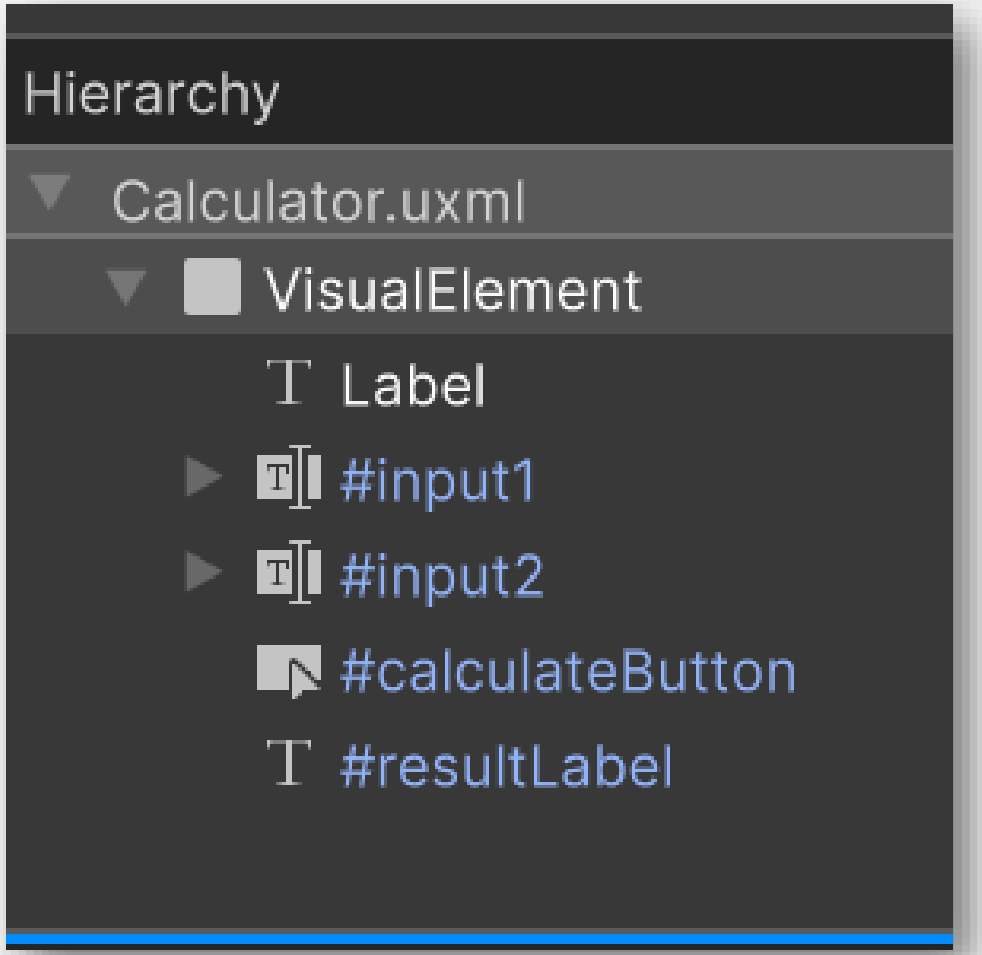
Unityอนุญาตให้คุณใช้ UI Toolkit และ Unity UI ร่วมกันได้ แต่ควรคำนึงถึงความแตกต่างของ API และระบบในการจัดการ

ตัวอย่าง

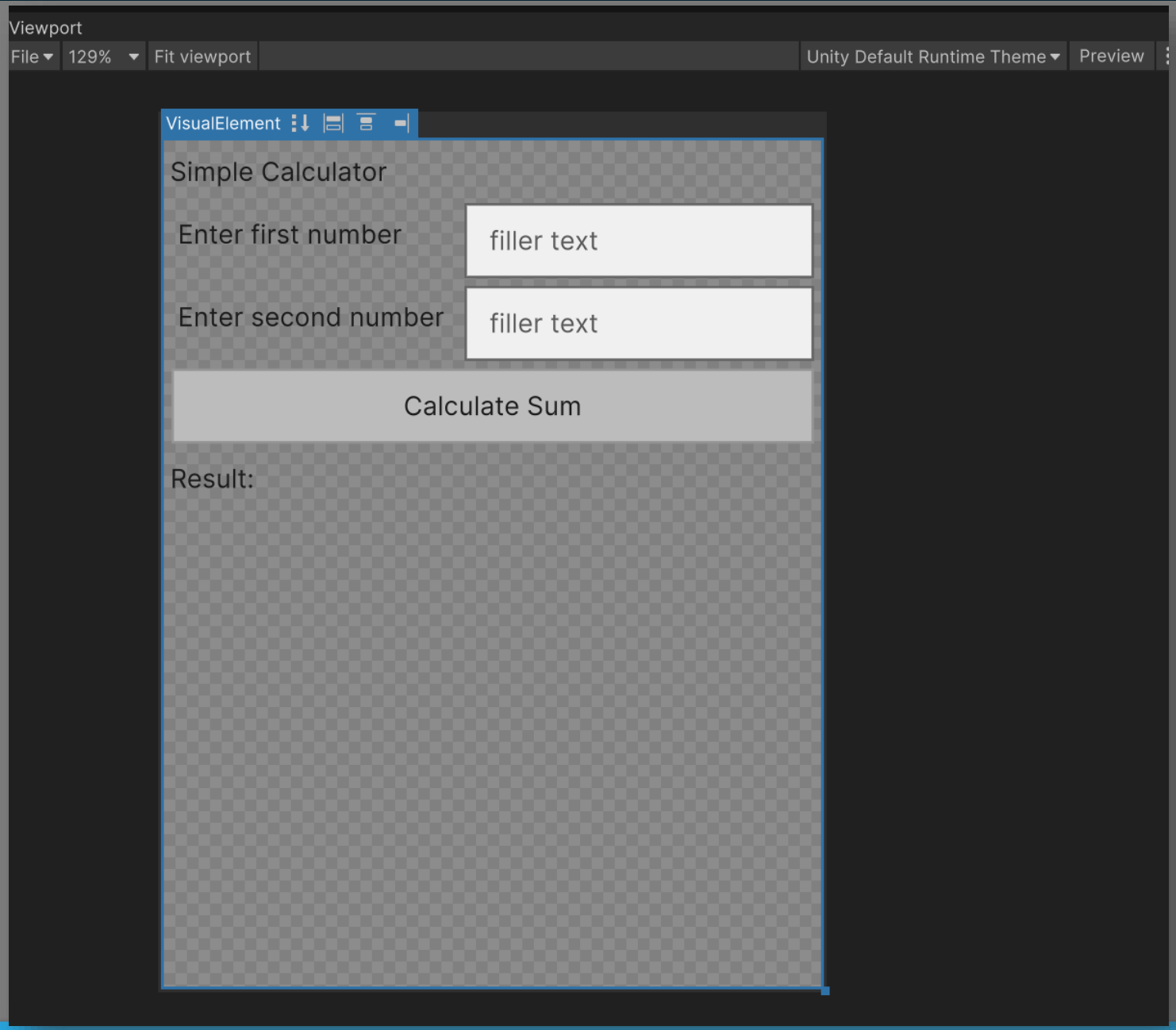
การใช้ UI Toolkit เพื่อรับค่าตัวเลข 2 ชุดจากผู้ใช้ และคำนวณผลรวมหลังจากกดปุ่ม

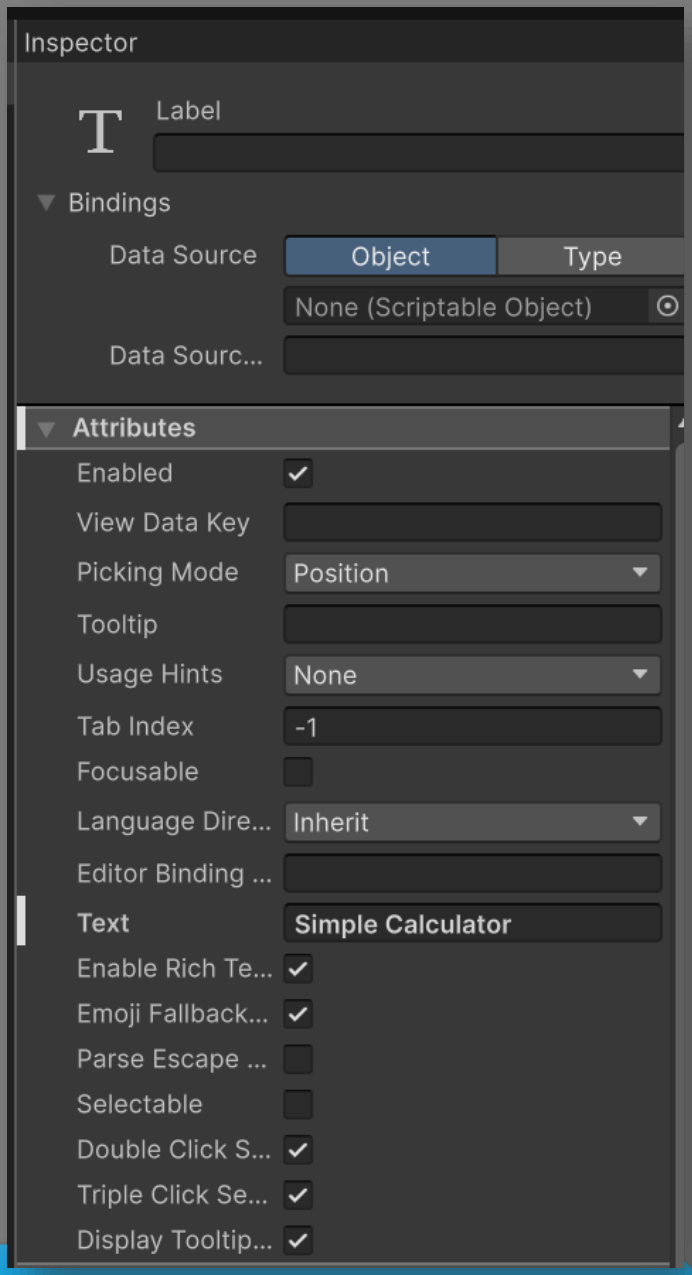

Game
Display 1
WXGA (1366x768)
Scale
0.85x
Play Focused
Simple Calculator
Enter first number
10
Enter second number
5
Calculate Sum
Result: 15

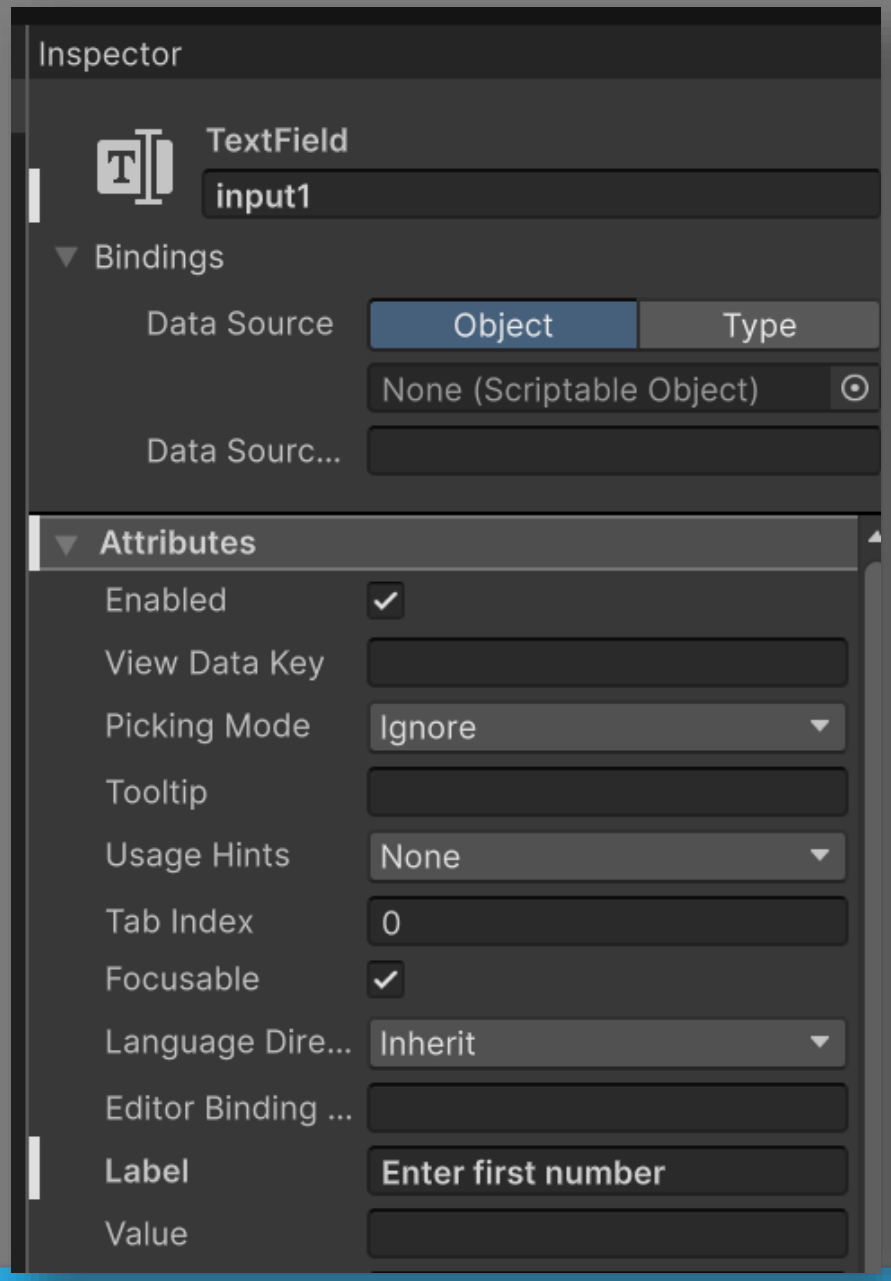
1 สร้างไฟล์ UXML

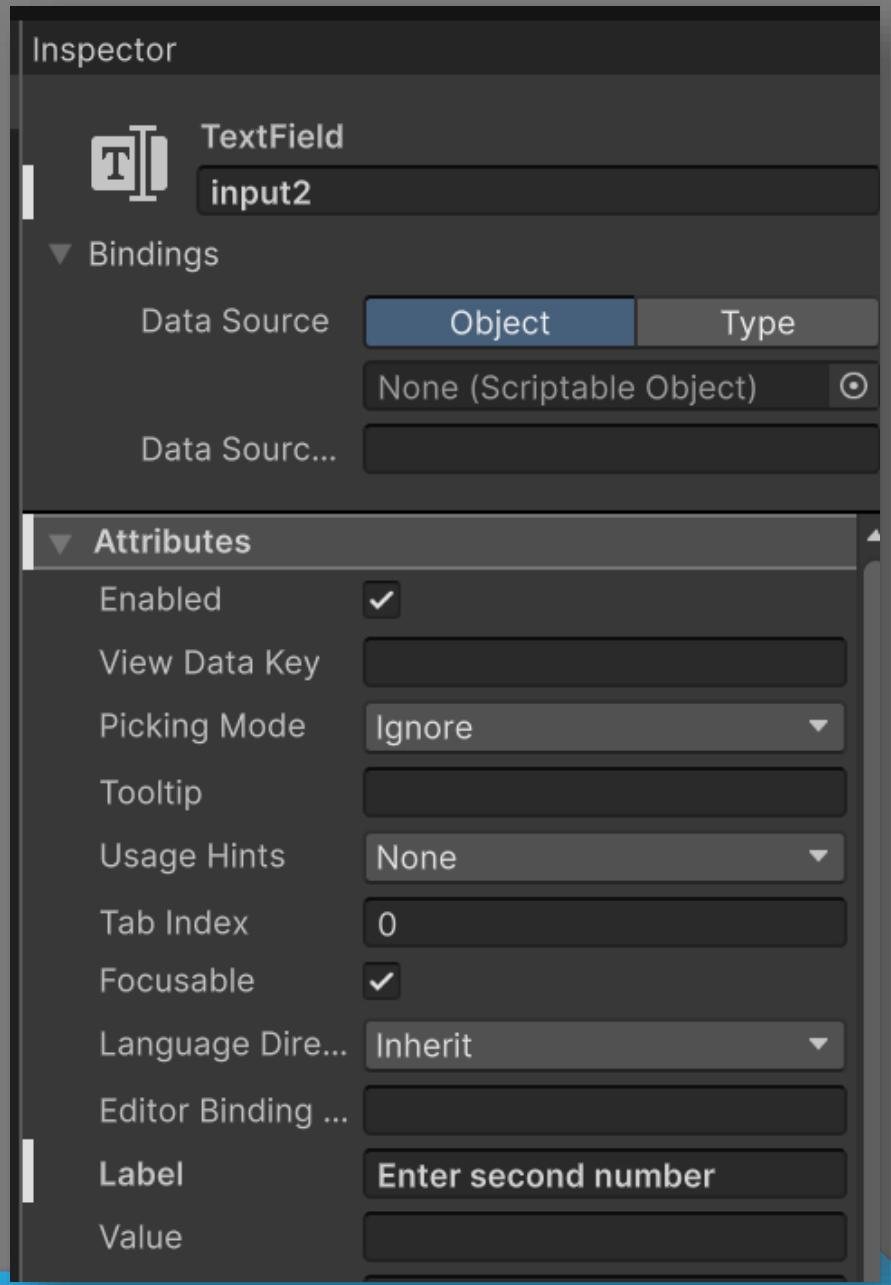


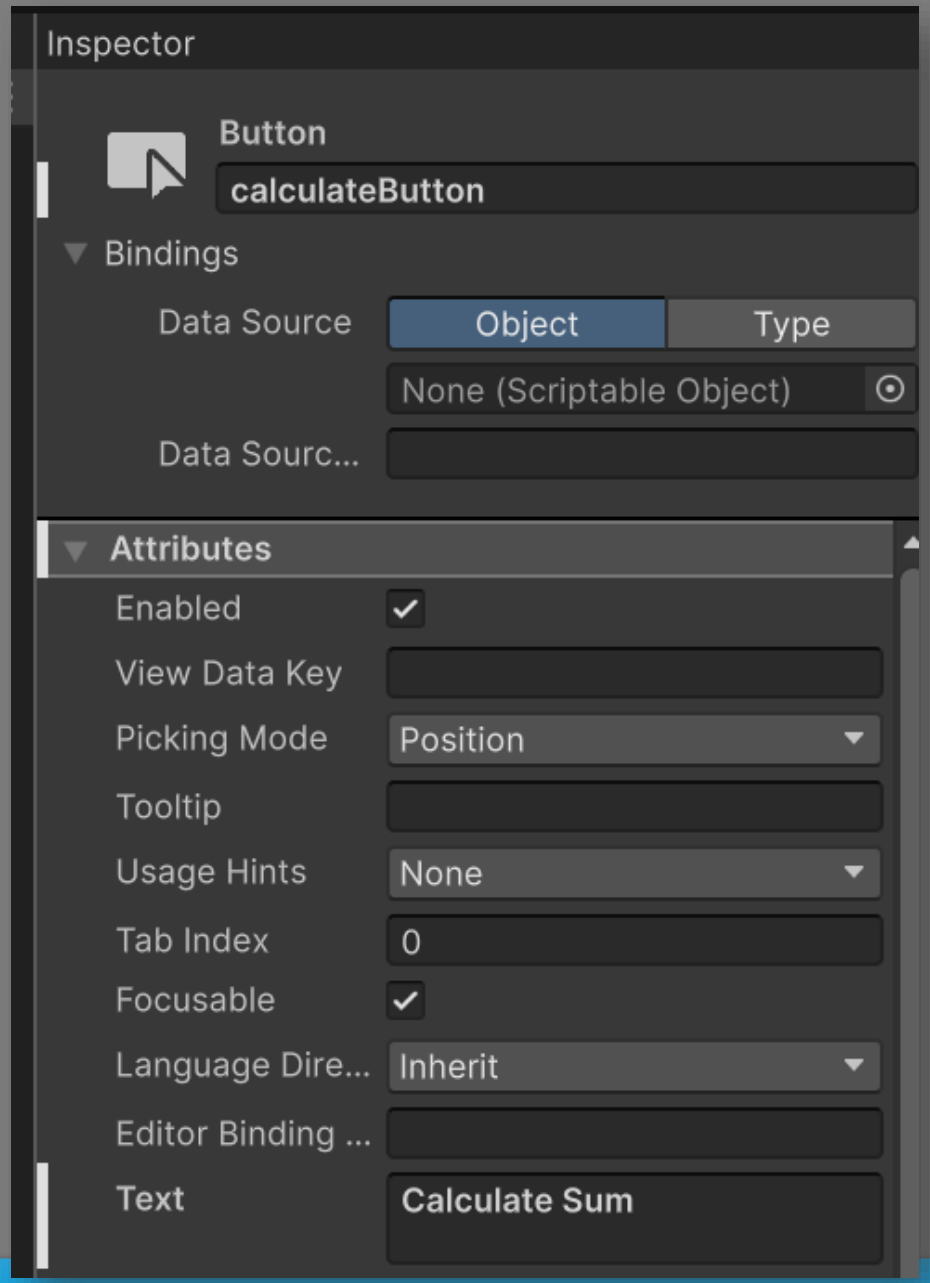
- สร้างไฟล์ Calculator.uxml ด้วยโครงสร้างต่อไปนี้

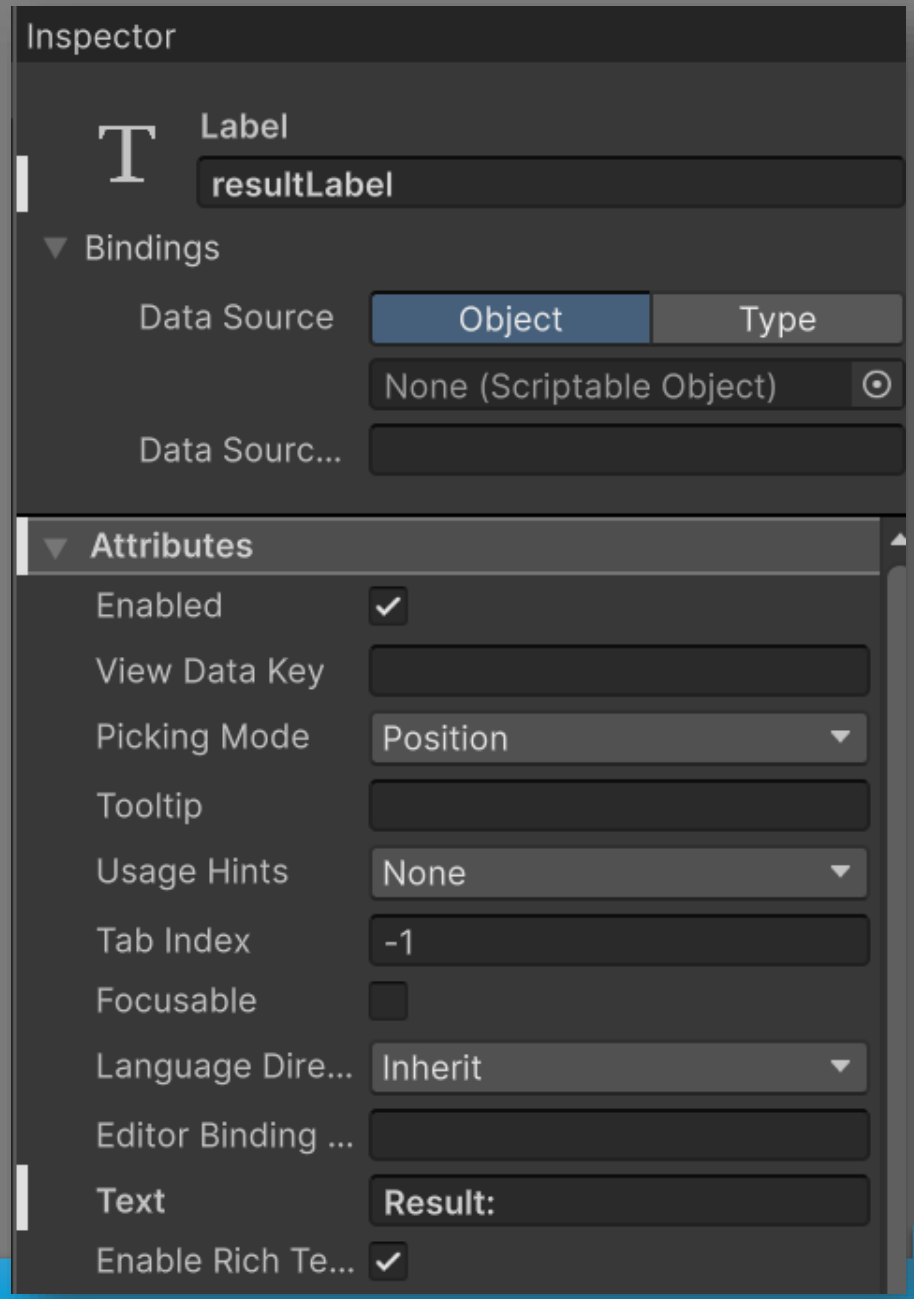

Viewport
File
129%
Fit viewport
Unity Default Runtime Theme
Preview
VisualElement
Simple Calculator
Enter first number
filler text
Enter second number
filler text
Calculate Sum
Result:

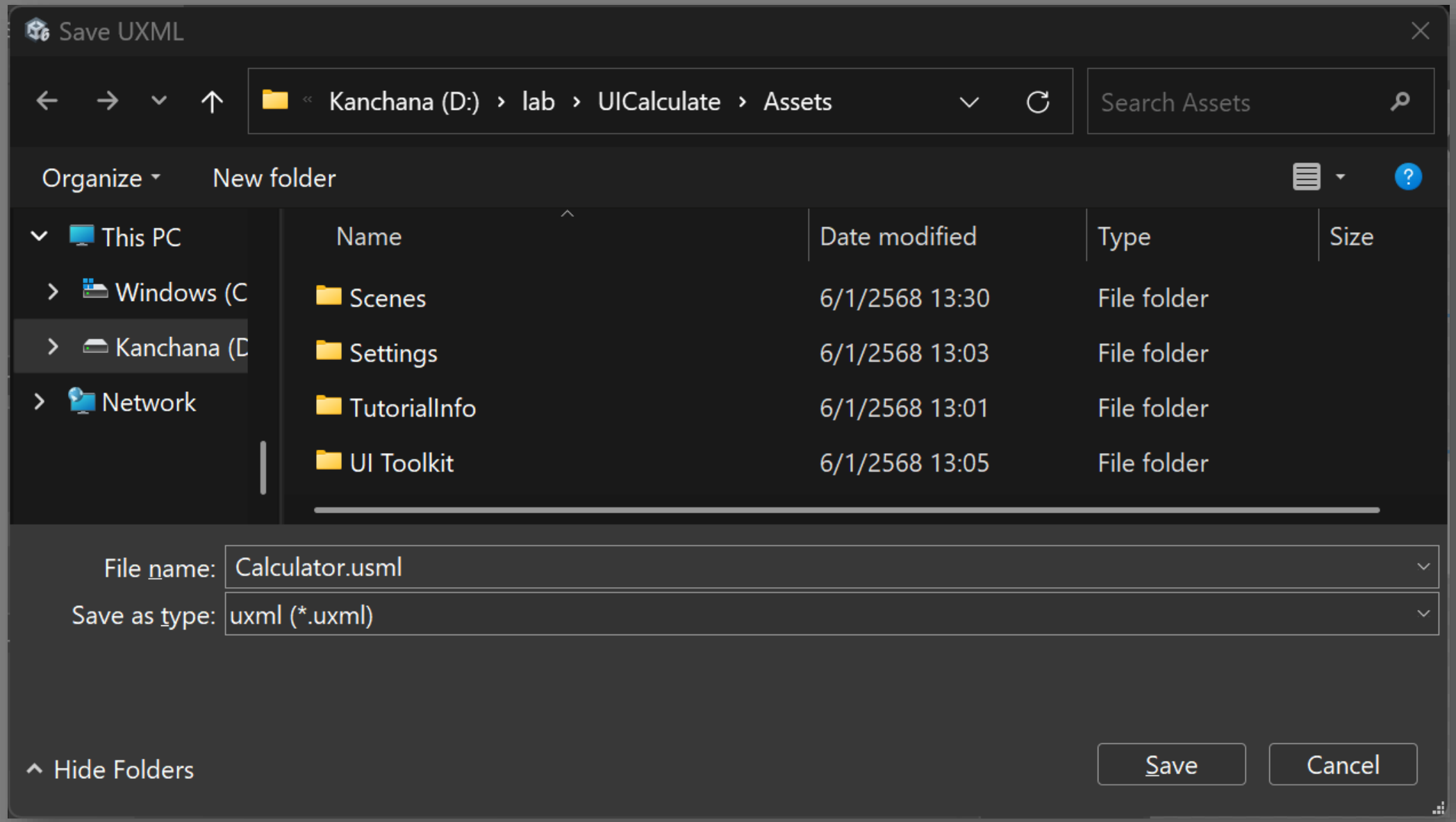

Inspector
Label
Bindings
Data Source
Object
Type
None (Scriptable Object)
Data Sourc...
Attributes
Enabled
View Data Key
Picking Mode
Position
Tooltip
Usage Hints
None
Tab Index
-1
Focusable
Language Dire...
Inherit
Editor Binding ...
Text
Simple Calculator
Enable Rich Te...
Emoji Fallback...
Parse Escape ...
Selectable
Double Click S...
Triple Click Se...
Display Tooltip...


Inspector
TextField
input1
Bindings
Data Source
Object
Type
None (Scriptable Object)
Data Sourc...
Attributes
Enabled
View Data Key
Picking Mode
Ignore
Tooltip
Usage Hints
None
Tab Index
0
Focusable
Language Dire...
Inherit
Editor Binding ...
Label
Enter first number
Value


Inspector
TextField
input2
Bindings
Data Source
Object
Type
None (Scriptable Object)
Data Sourc...
Attributes
Enabled
View Data Key
Picking Mode
Ignore
Tooltip
Usage Hints
None
Tab Index
0
Focusable
Language Dire...
Inherit
Editor Binding ...
Label
Enter second number
Value

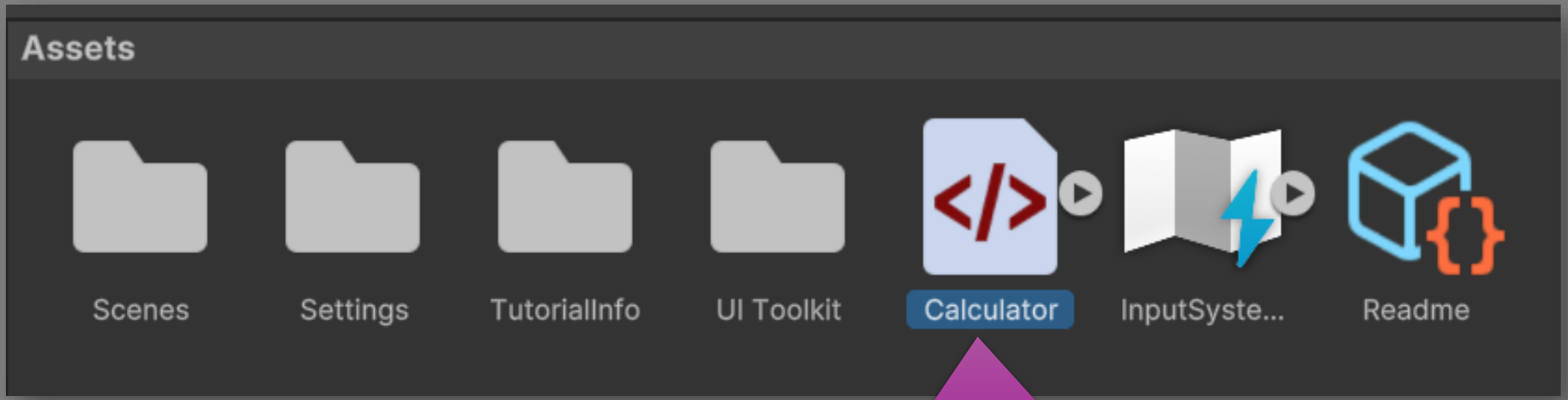

Inspector
Button
calculateButton
Bindings
Data Source
Object
Type
None (Scriptable Object)
Data Sourc...
Attributes
Enabled
View Data Key
Picking Mode
Position
Tooltip
Usage Hints
None
Tab Index
0
Focusable
Language Dire...
Inherit
Editor Binding ...
Text
Calculate Sum


Inspector
Label
resultLabel
Bindings
Data Source
Object
Type
None (Scriptable Object)
Data Sourc...
Attributes
Enabled
View Data Key
Picking Mode
Position
Tooltip
Usage Hints
None
Tab Index
-1
Focusable
Language Dire...
Inherit
Editor Binding ...
Text
Result:
Enable Rich Te...


Save UXML
Kanchana (D:) › lab › UICalculate › Assets
Search Assets
Organize
New folder
This PC
Windows (C
Kanchana (D
Network
Name
Date modified
Type
Size
Scenes
6/1/2568 13:30
File folder
Settings
6/1/2568 13:03
File folder
TutorialInfo
6/1/2568 13:01
File folder
UI Toolkit
6/1/2568 13:05
File folder
File name:
Calculator.usml
Save as type:
uxml (*.uxml)
Hide Folders
Save
Cancel

Calculator.uxml

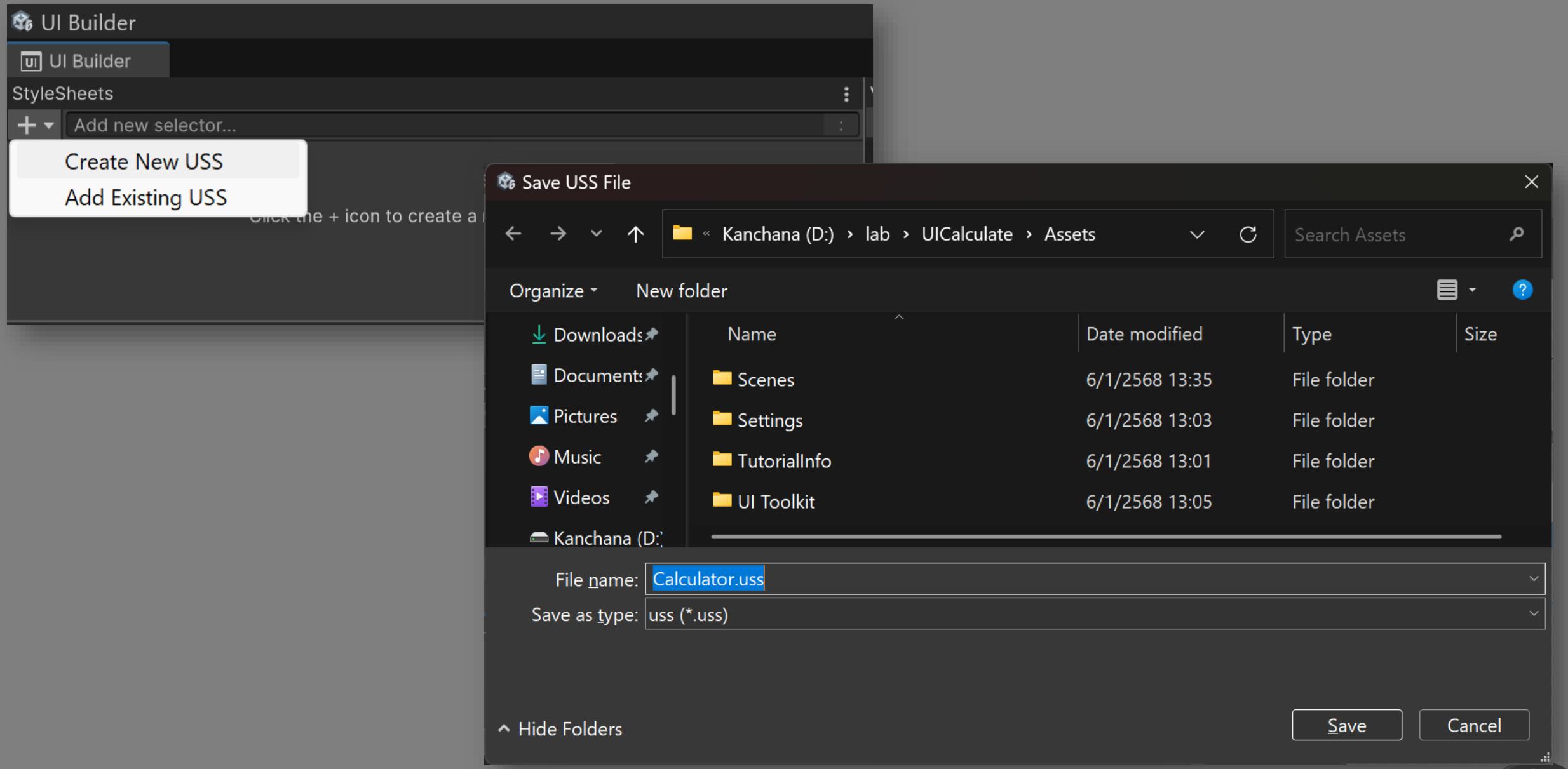

```
<ui:UXML xmlns:ui="UnityEngine.UIElements">
    <ui:VisualElement class="root">
        <ui:Label text="Simple Calculator" class="title" />
        <ui:TextField name="input1" label="Enter first number" />
        <ui:TextField name="input2" label="Enter second number" />
        <ui:Button name="calculateButton" text="Calculate Sum" />
        <ui:Label name="resultLabel" text="Result: " class="result" />
    </ui:VisualElement>
</ui:UXML>
```

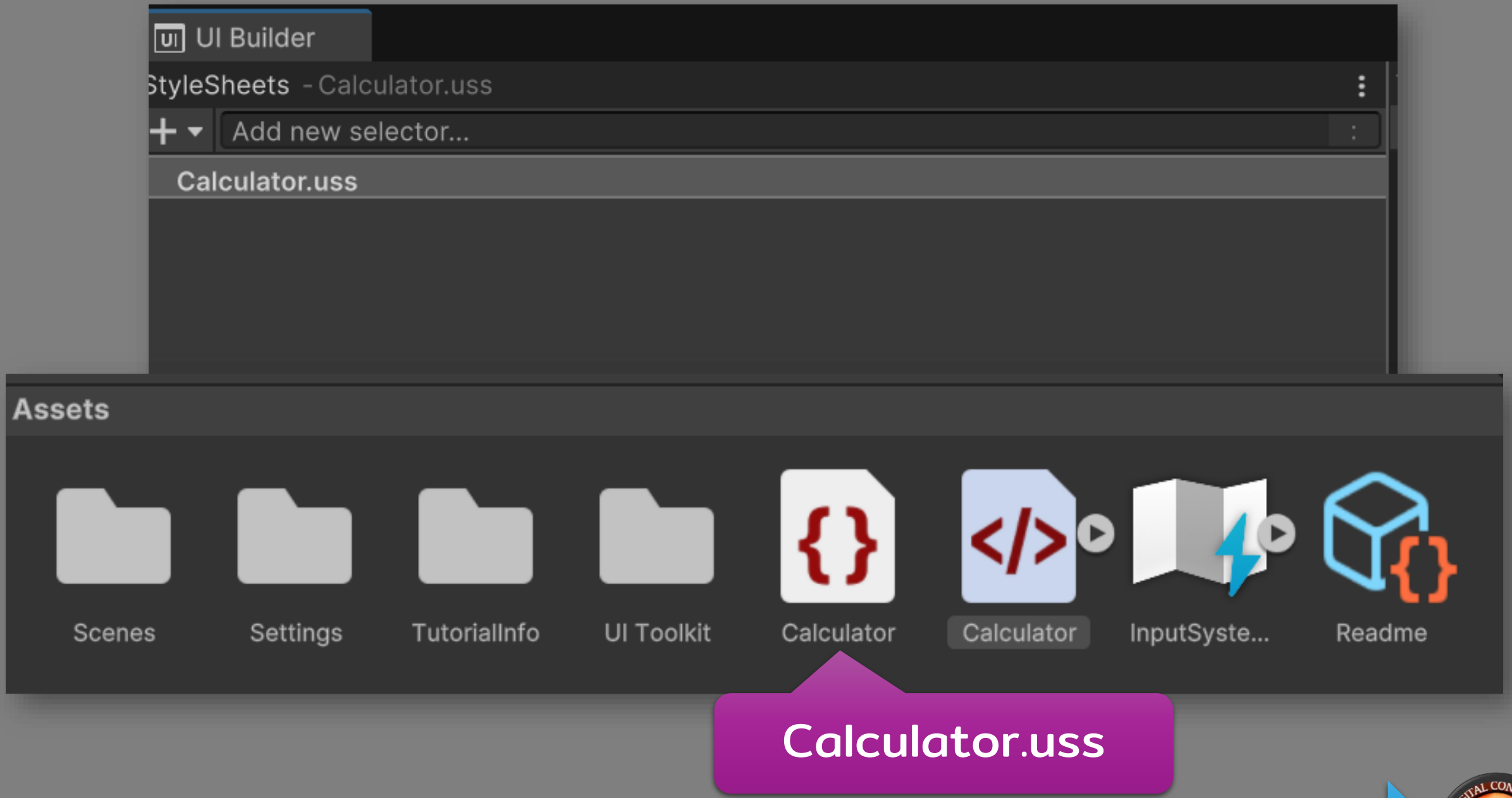


Calculator.uxml

2 สร้างไฟล์ USS

- สร้างไฟล์ Calculator.uss เพื่อจัดสไตล์ UI

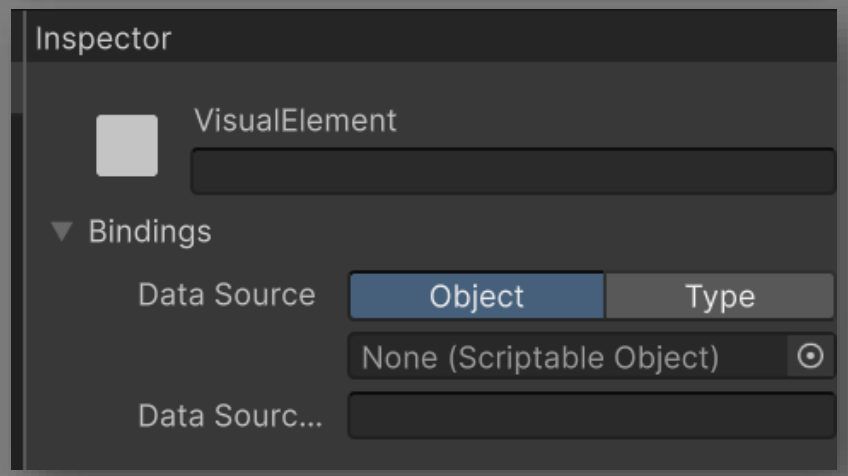

UI Builder
UI Builder
StyleSheets
Add new selector...
Create New USS
Add Existing USS
Save USS File
Kanchana (D:) › lab › UICalculate › Assets
Search Assets
Organize
New folder
Downloads
Documents
Pictures
Music
Videos
Kanchana (D:)
Name
Date modified
Type
Size
Scenes
6/1/2568 13:35
File folder
Settings
6/1/2568 13:03
File folder
TutorialInfo
6/1/2568 13:01
File folder
UI Toolkit
6/1/2568 13:05
File folder
File name:
Calculator.uss
Save as type:
uss (*.uss)
Hide Folders
Save
Cancel

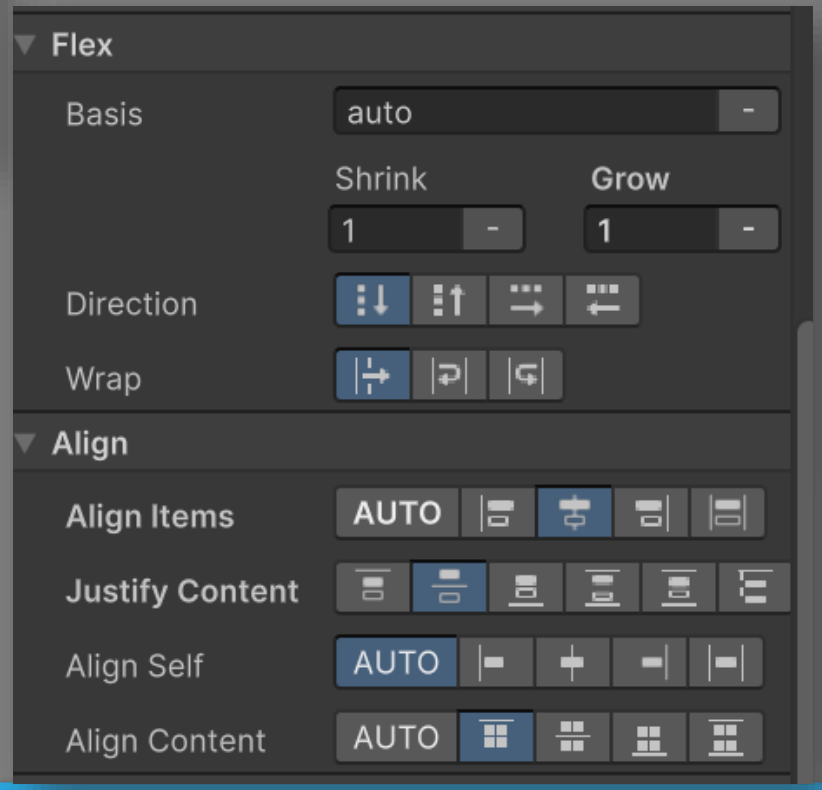

UI Builder
StyleSheets - Calculator.uss
Add new selector...
Calculator.uss
Assets
Scenes
Settings
TutorialInfo
UI Toolkit
Calculator
Calculator
InputSyste...
Readme
Calculator.uss

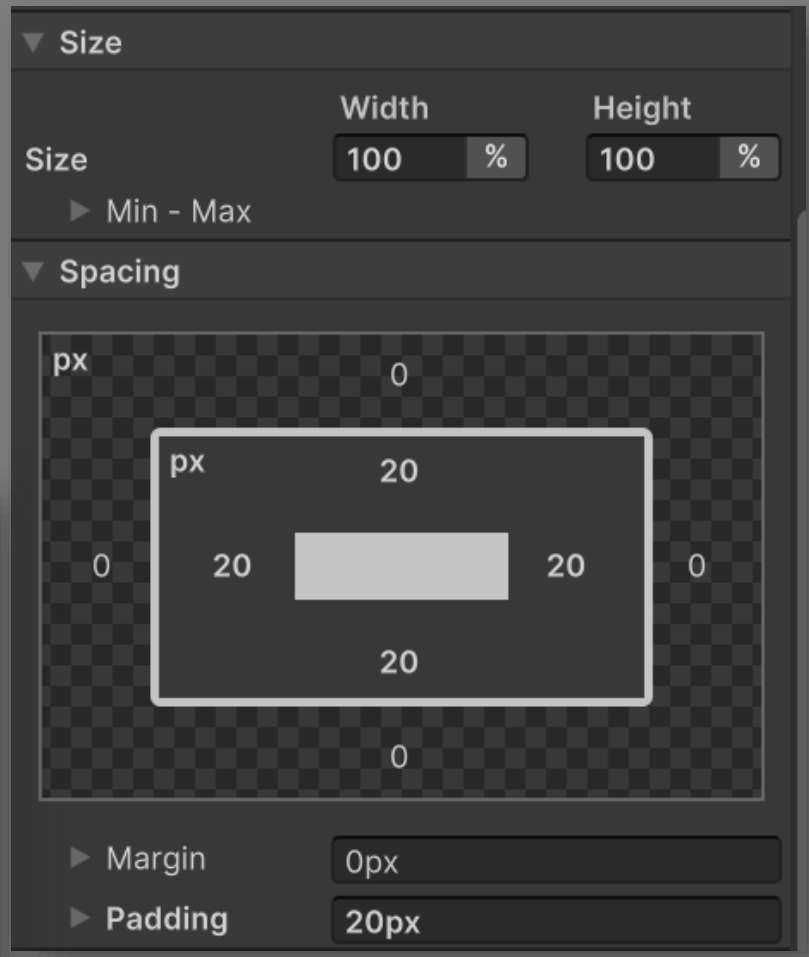
Calculator.uss


```
.root {
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 100%;
    height: 100%;
    padding: 20px;
}
.title {
    font-size: 24px;
    margin-bottom: 20px;
    color: white;
}
```

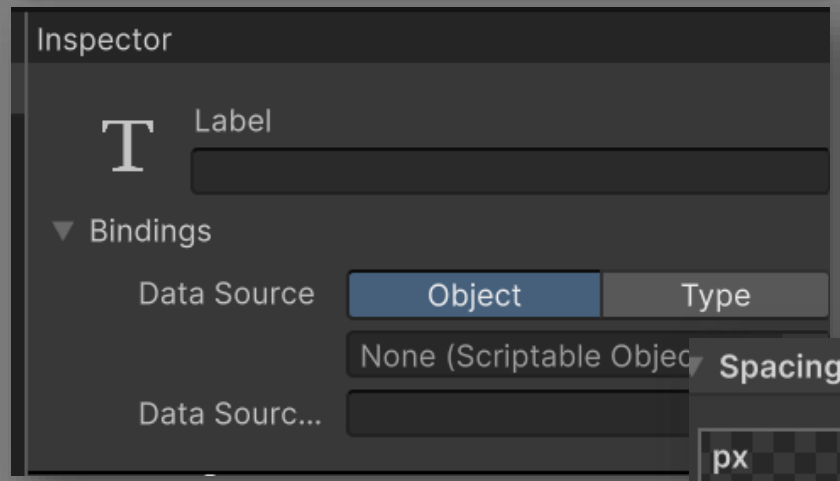
VisualElement .root

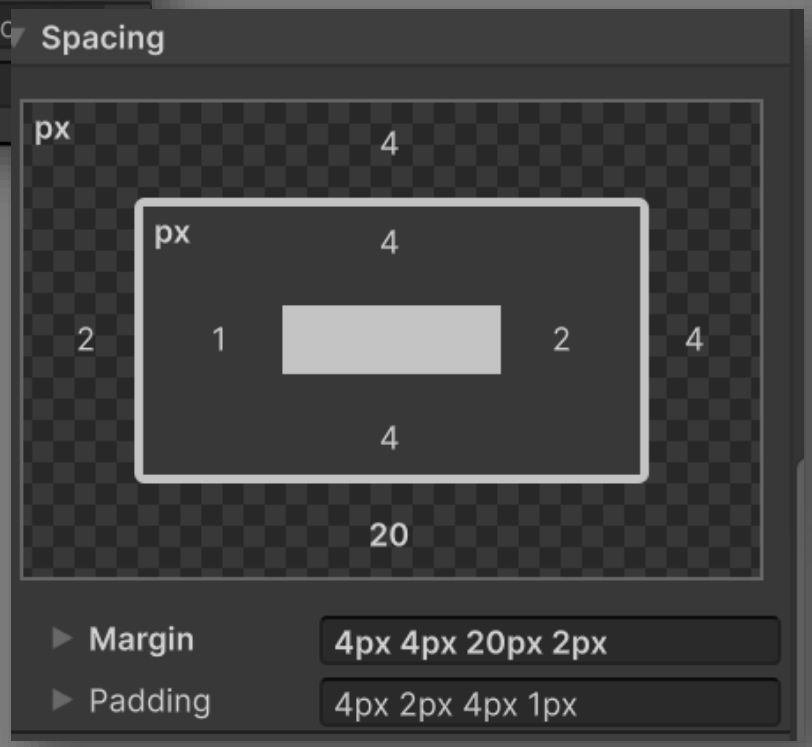


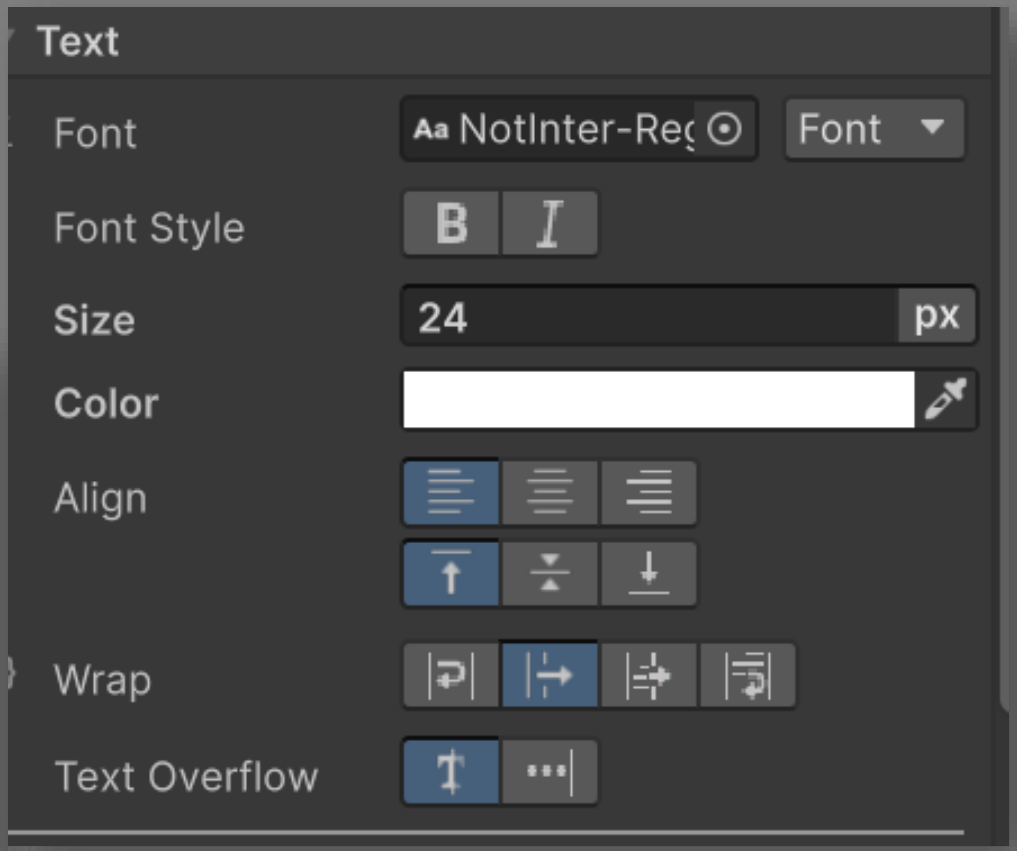




Label .title


Inspector
Label
Bindings
Data Source
Object
Type
None (Scriptable Objec
Data Sourc...

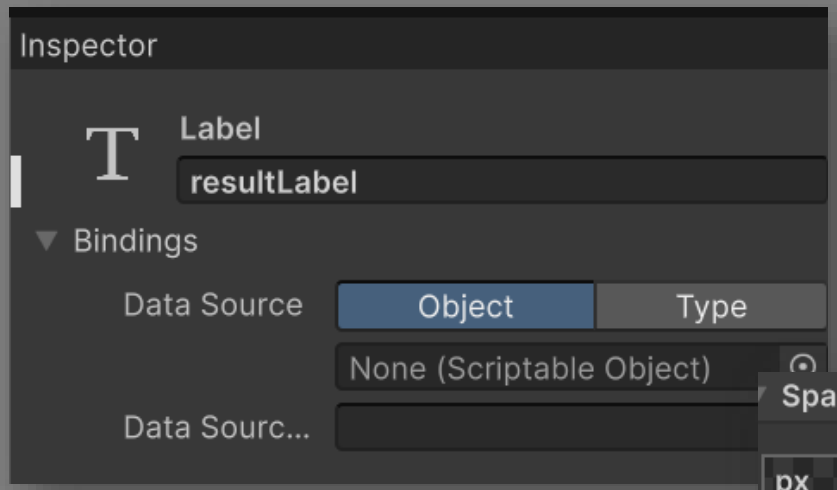

Spacing
px
4
px
4
2
1
2
4
4
20
Margin
4px 4px 20px 2px
Padding
4px 2px 4px 1px

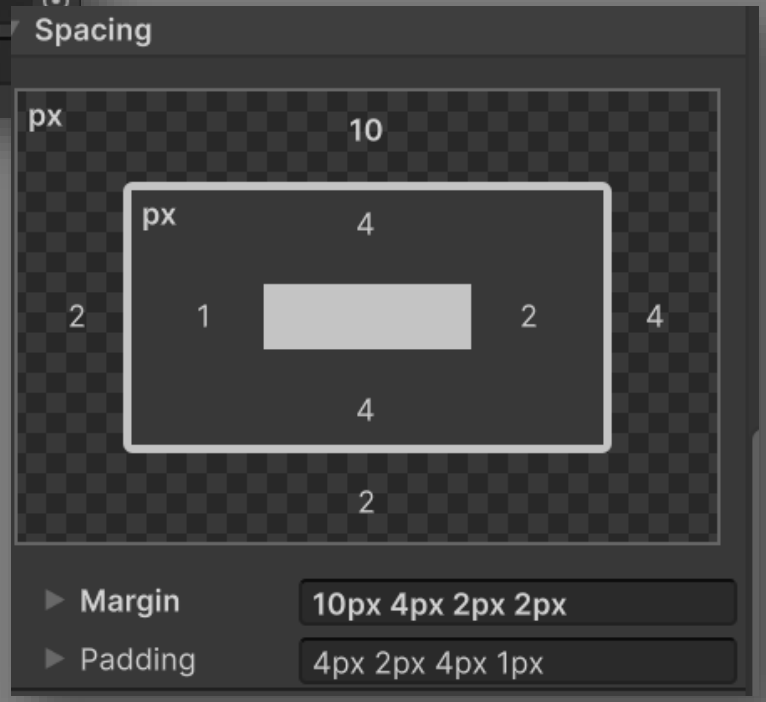

Text
Font
NotInter-Reg
Font
Font Style
B
I
Size
24
px
Color
Align
Wrap
Text Overflow

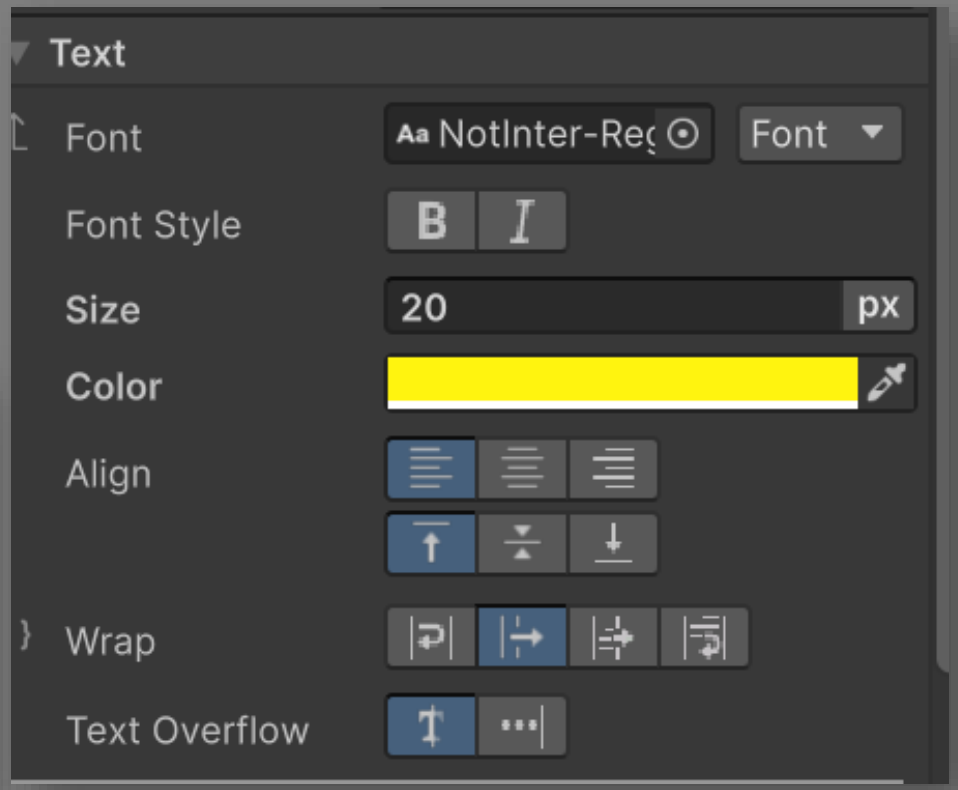
Calculator.uss


```
.result {
    font-size: 20px;
    margin-top: 10px;
    color: yellow;
}
TextField {
    margin-bottom: 10px;
    width: 300px;
}
```

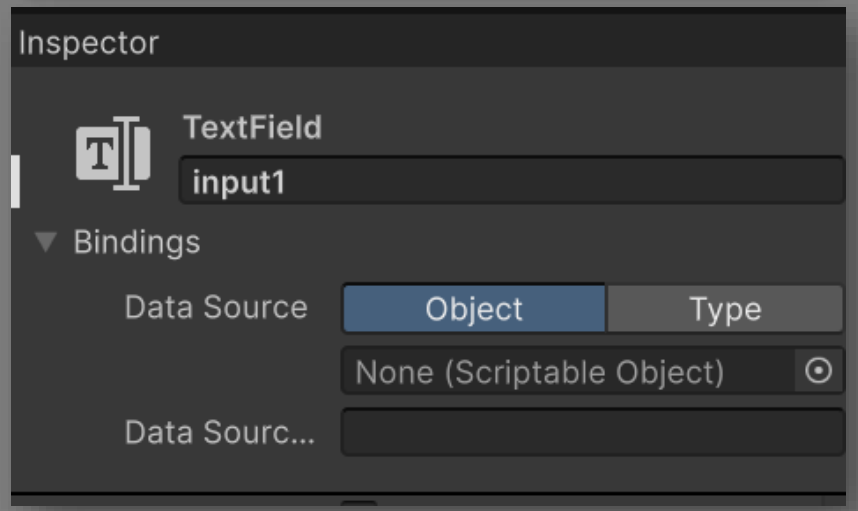
resultLabel .result

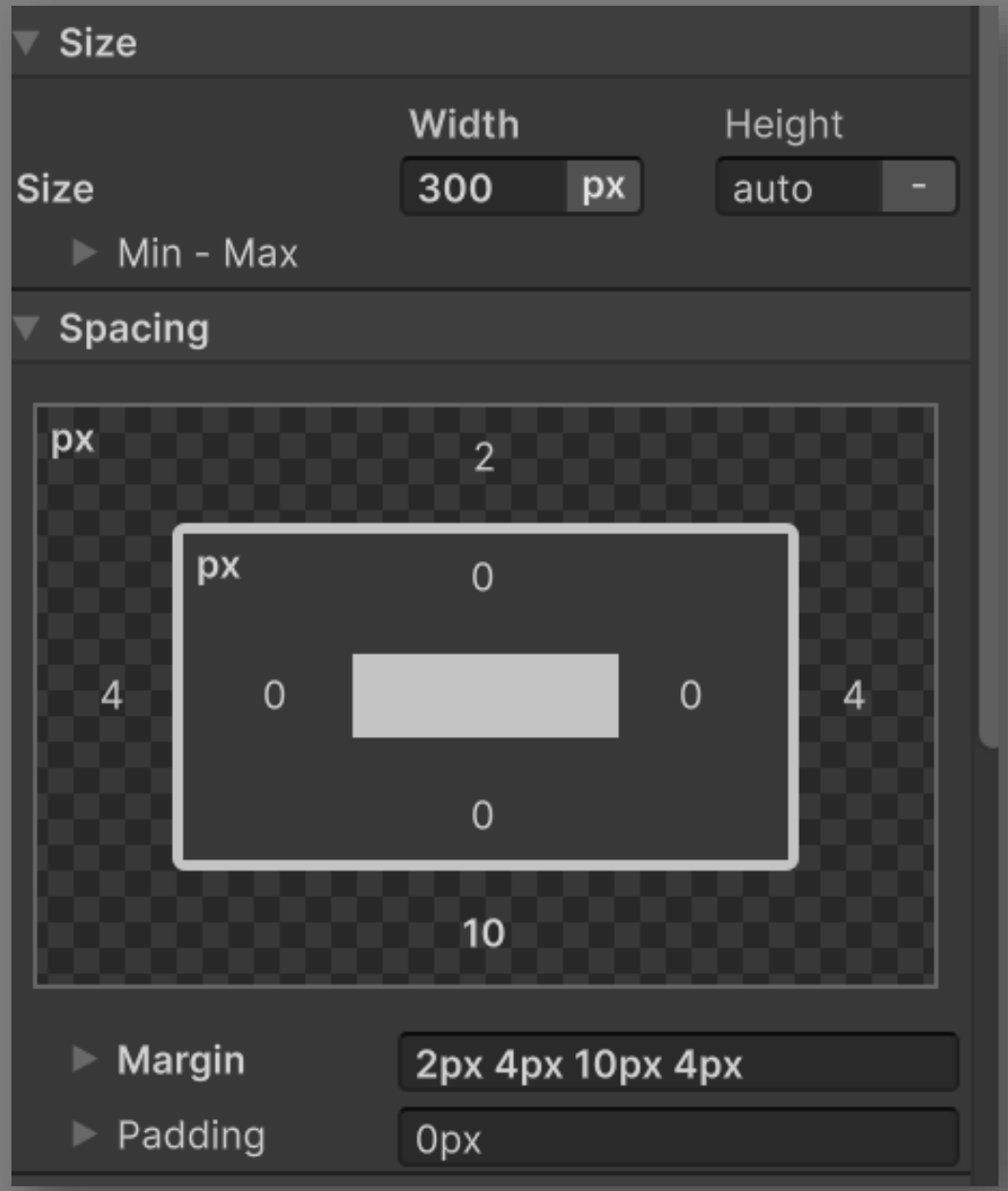




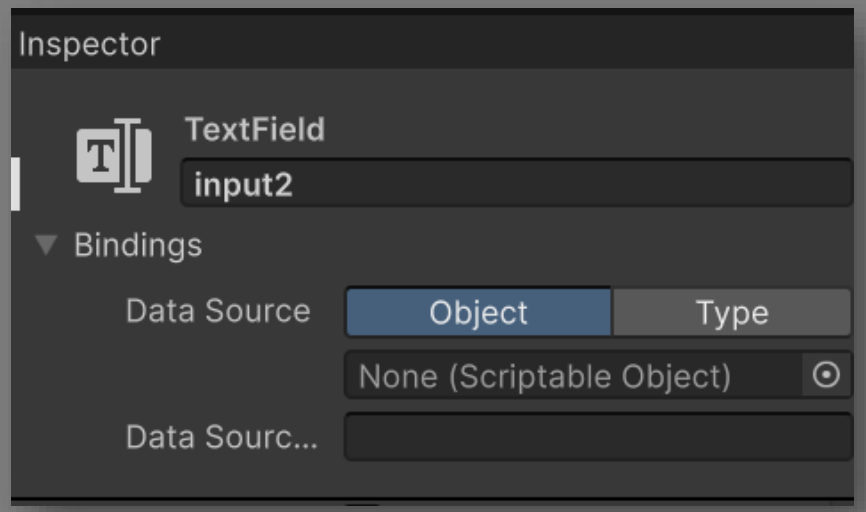


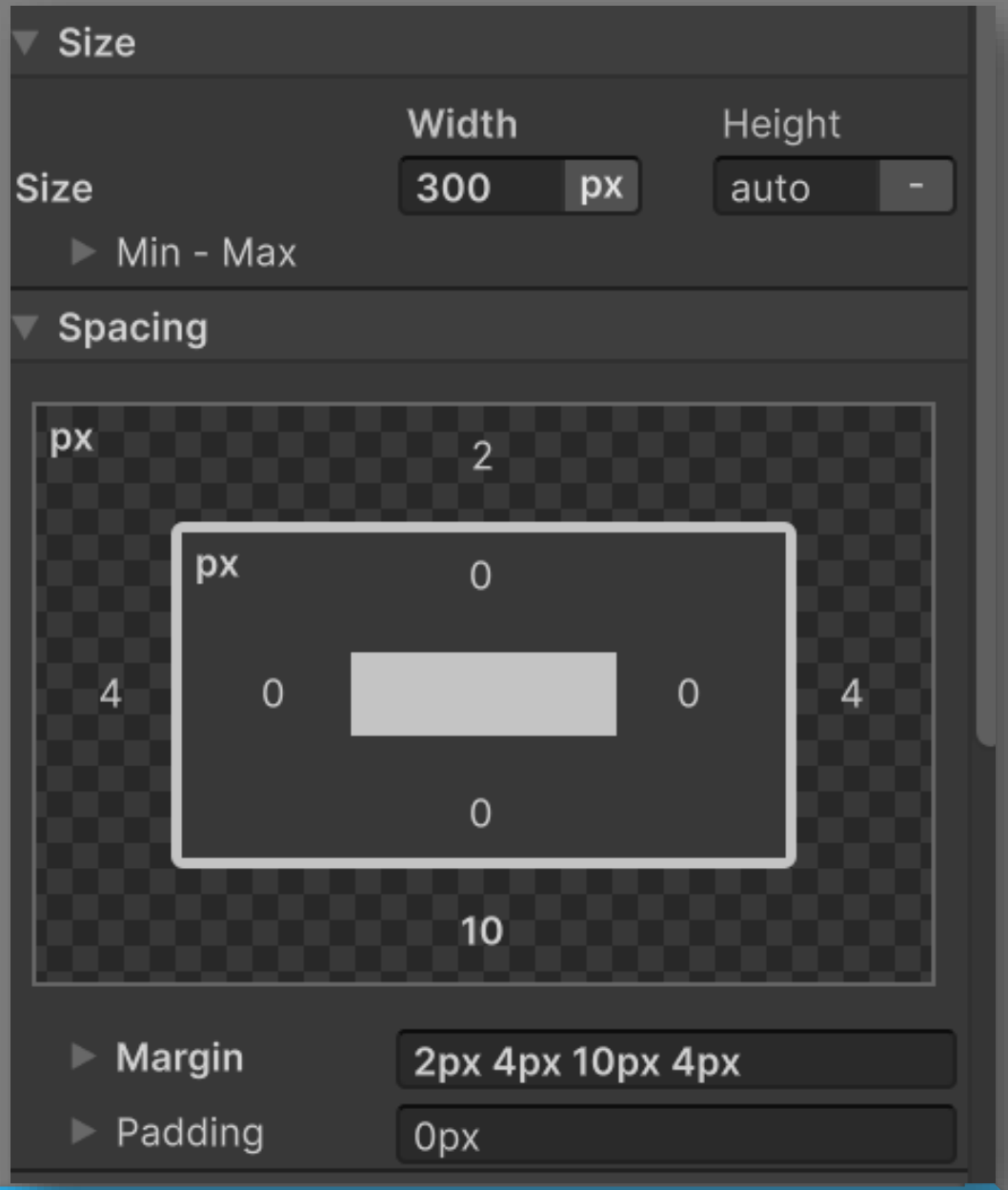
input1 TextField

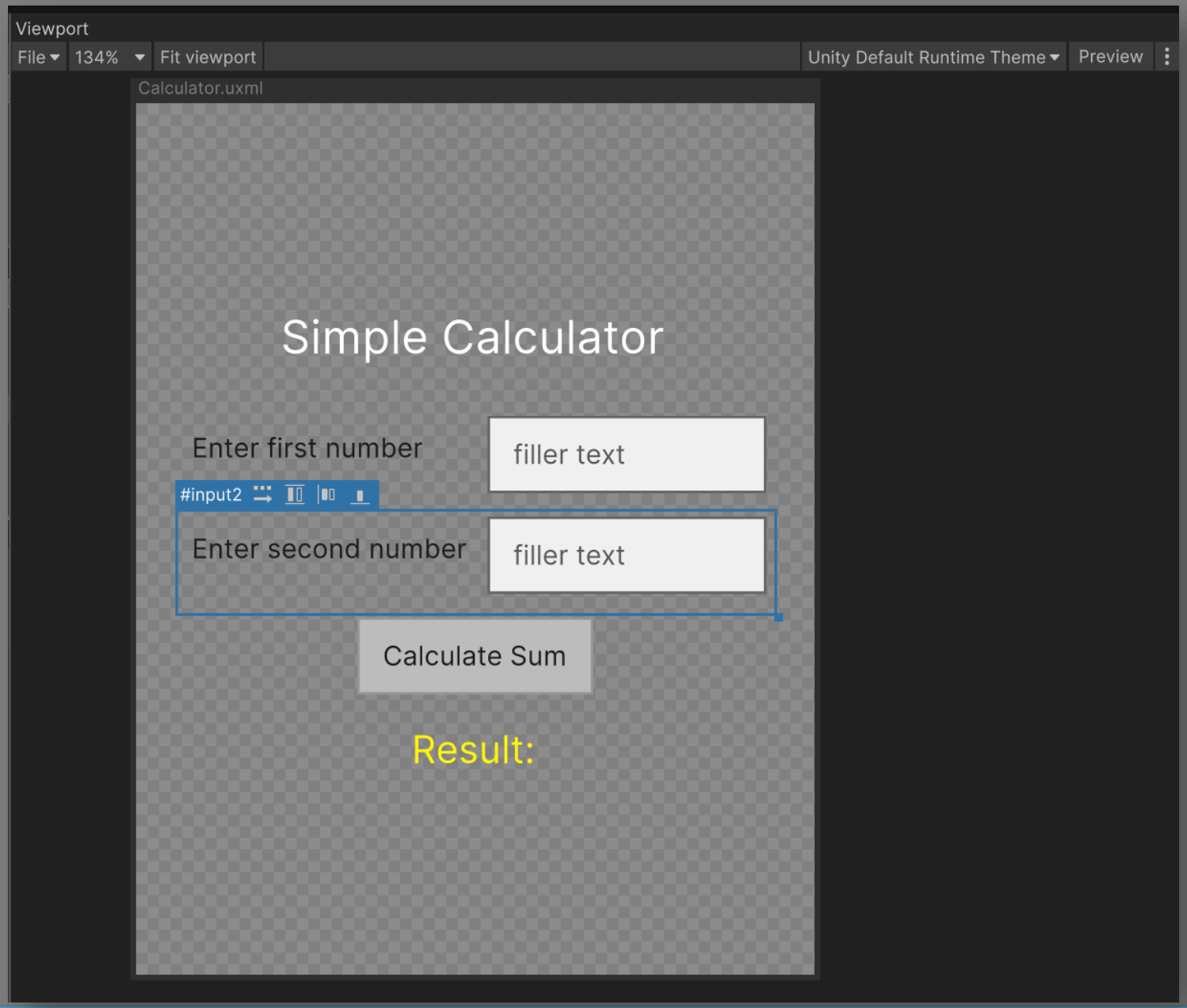




input2 TextField

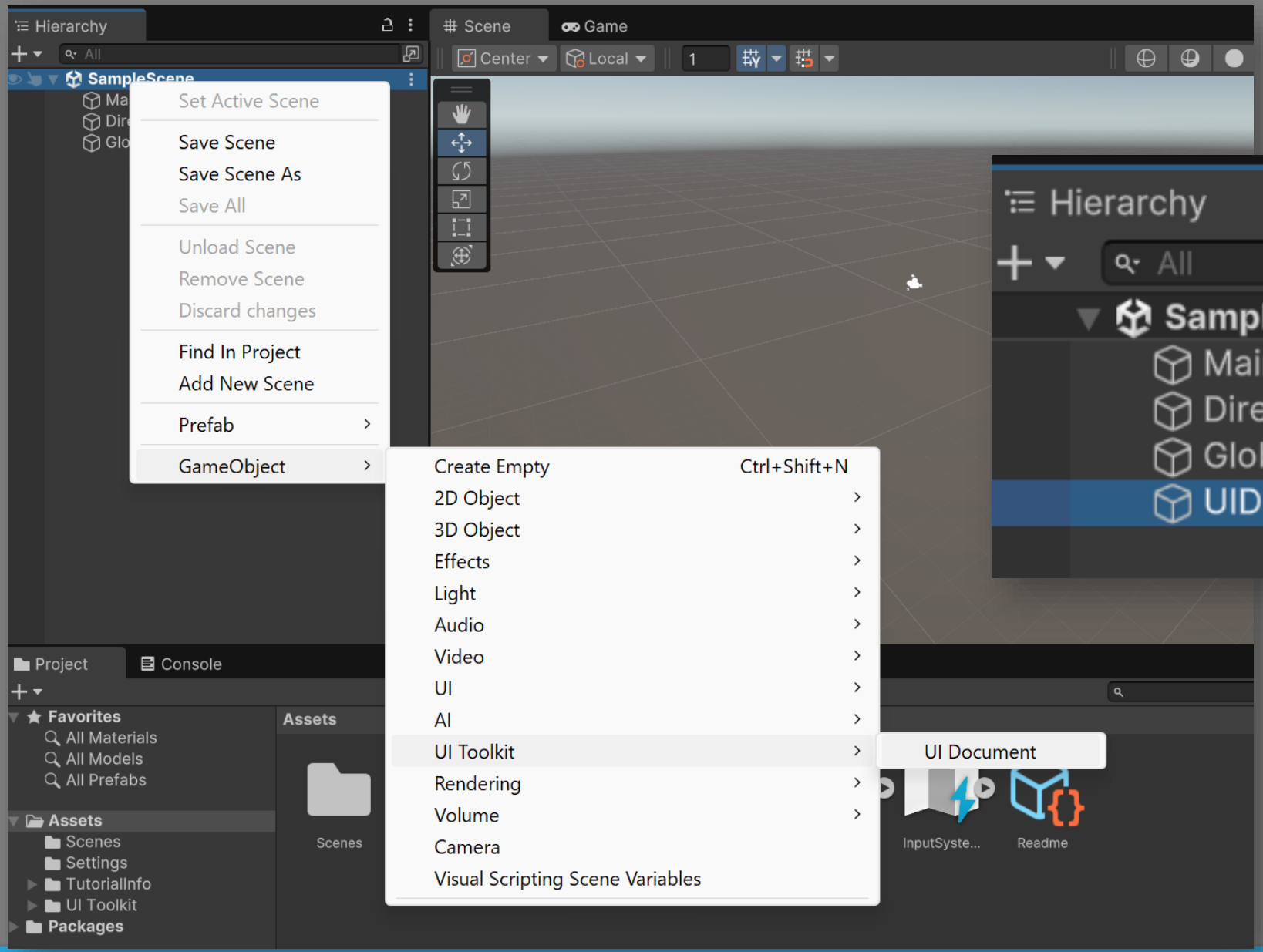


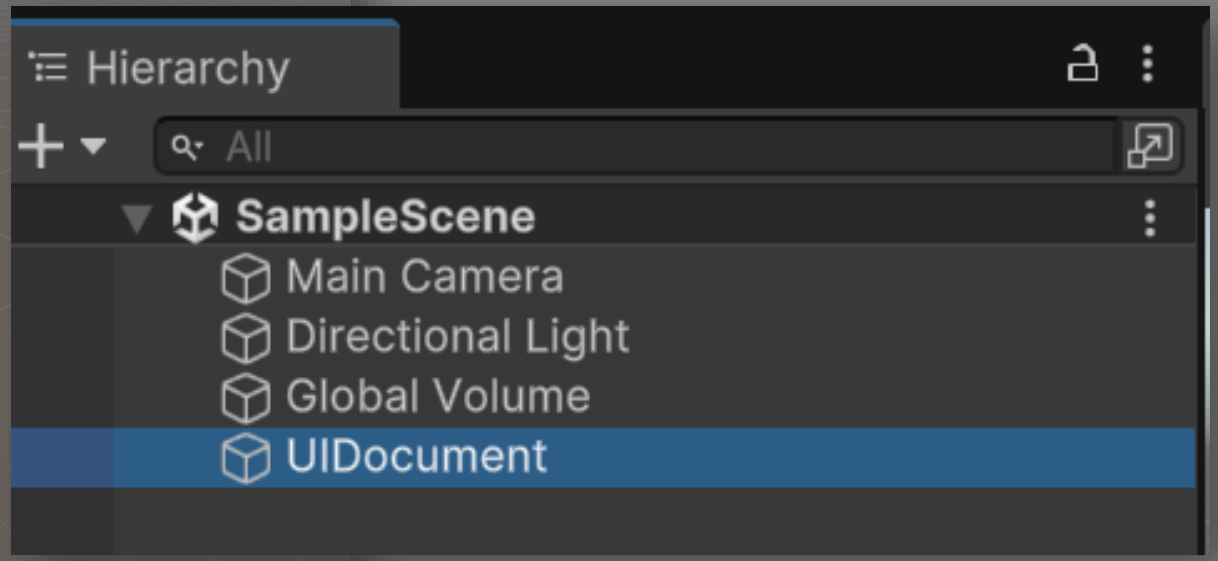


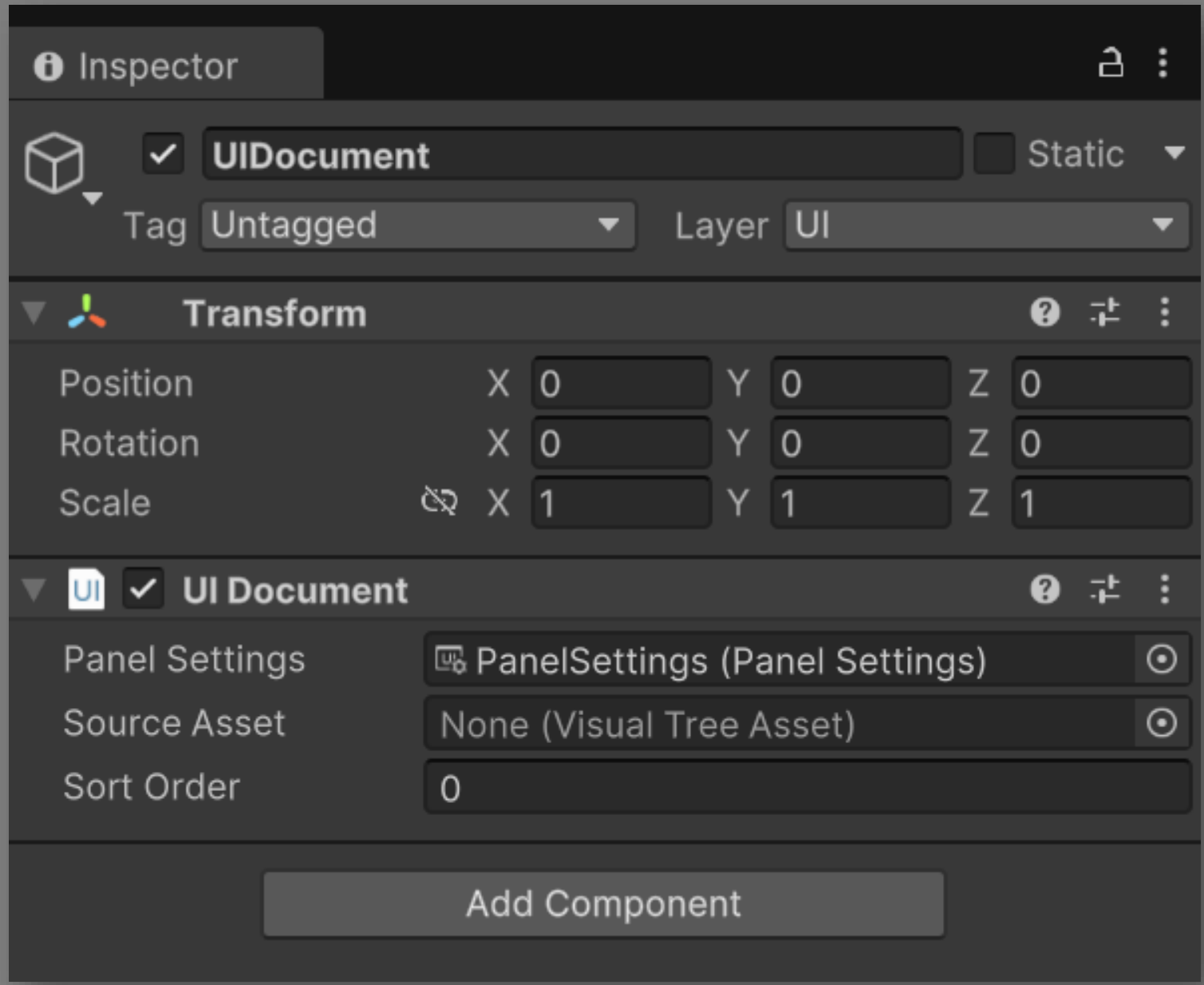

Viewport
File
134%
Fit viewport
Unity Default Runtime Theme
Preview
Calculator.uxml
Simple Calculator
Enter first number
filler text
#input2
Enter second number
filler text
Calculate Sum
Result:

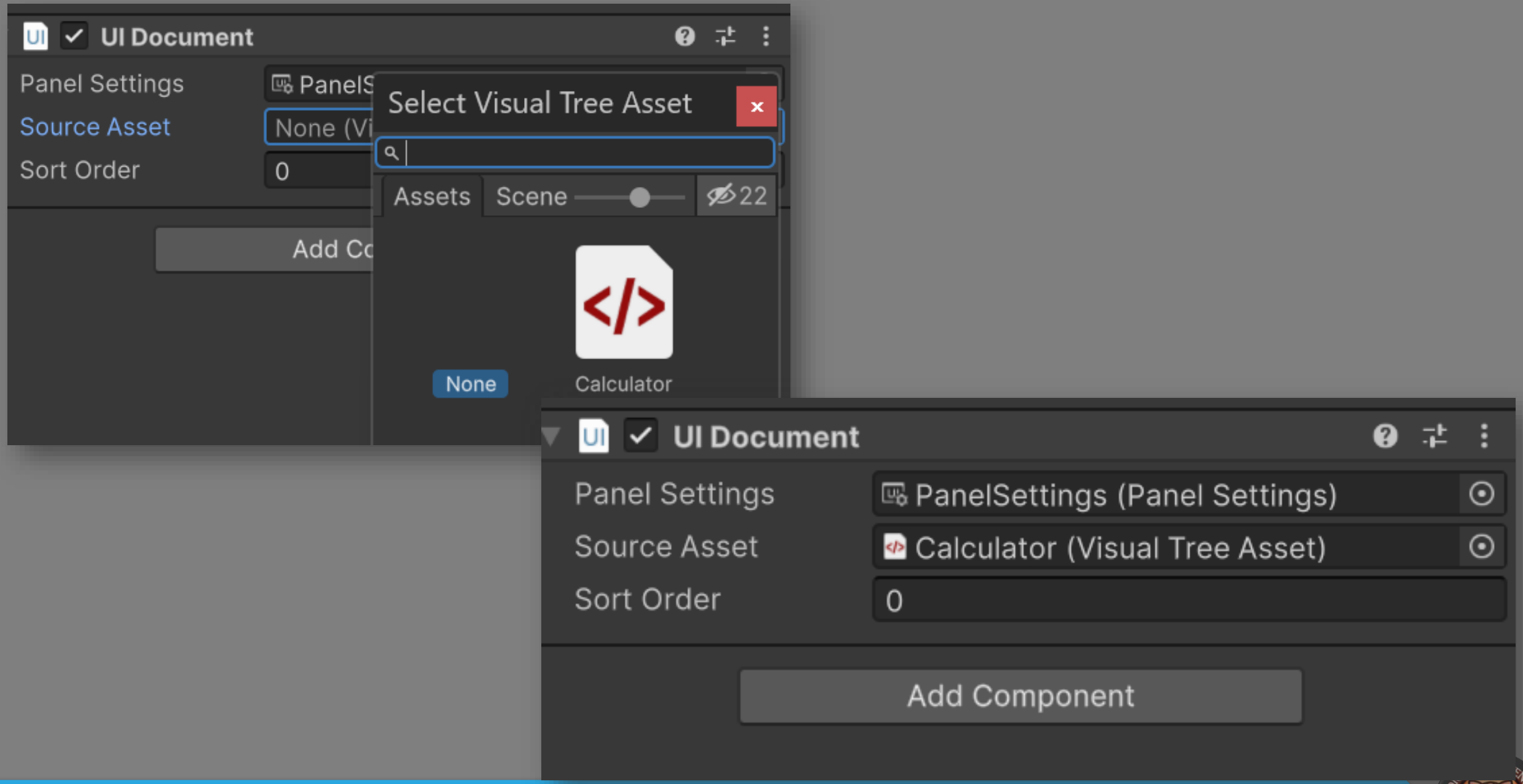
3 เชื่อมไฟล์ UXML และ USS กับ UIDocument

- สร้าง GameObject ใน Scene
- เพิ่มคอมโพเนนต์ UIDocument
- ตั้งค่าฟิลด์ Source Asset เป็นไฟล์ Calculator.uxml


Hierarchy
Scene
Game
Center
Local
SampleScene
Set Active Scene
Save Scene
Save Scene As
Save All
Unload Scene
Remove Scene
Discard changes
Find In Project
Add New Scene
Prefab
GameObject
Create Empty Ctrl+Shift+N
2D Object
3D Object
Effects
Light
Audio
Video
UI
AI
UI Toolkit
Rendering
Volume
Camera
Visual Scripting Scene Variables
UI Document
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Scenes
Settings
TutorialInfo
UI Toolkit
Packages
InputSyste...
Readme

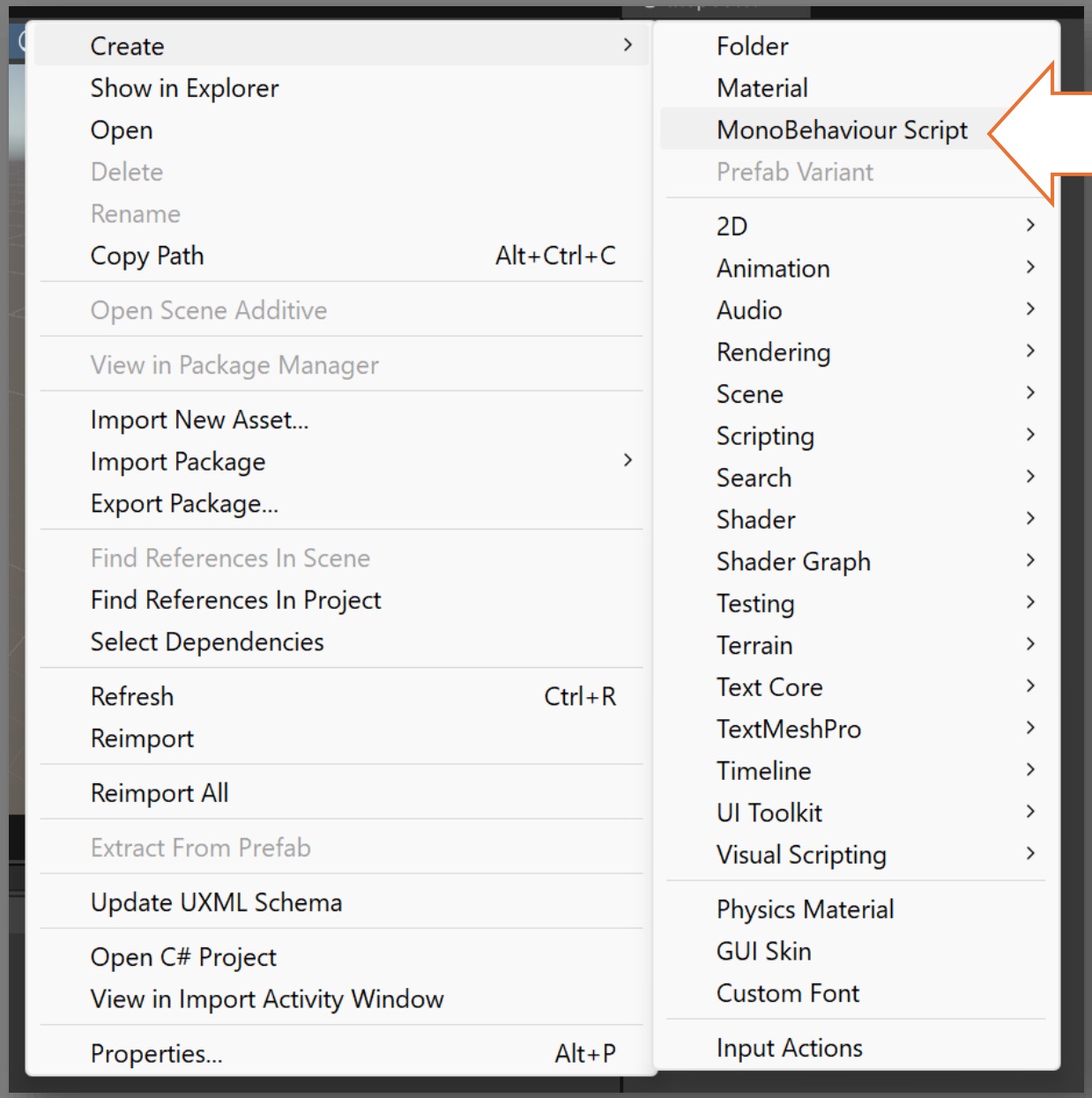

Hierarchy
All
SampleScene
Main Camera
Directional Light
Global Volume
UIDocument

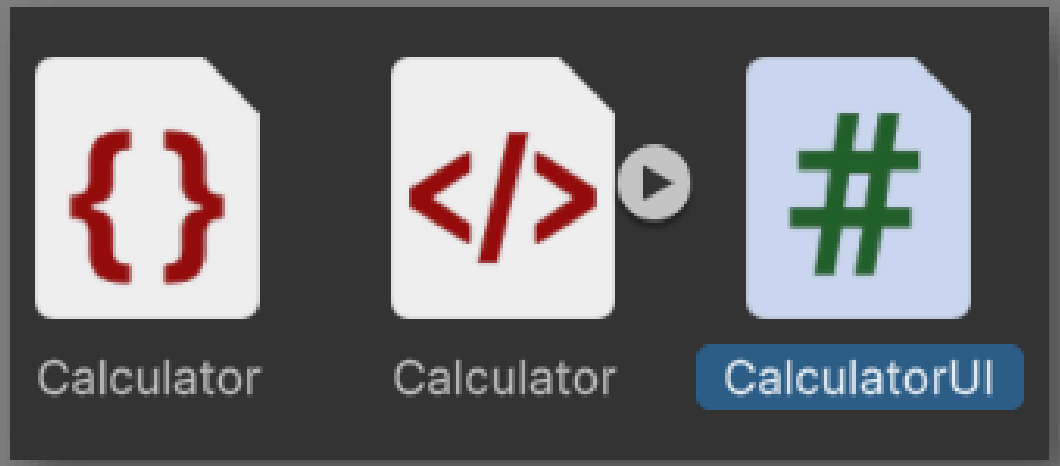

Inspector
UIDocument
Static
Tag
Untagged
Layer
UI
Transform
Position
X 0
Y 0
Z 0
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
UI Document
Panel Settings
PanelSettings (Panel Settings)
Source Asset
None (Visual Tree Asset)
Sort Order
0
Add Component


UI Document
Panel Settings
PanelS
Source Asset
None (Vi
Sort Order
0
Add Co
Select Visual Tree Asset
Assets
Scene
22
None
Calculator
UI Document
Panel Settings
PanelSettings (Panel Settings)
Source Asset
Calculator (Visual Tree Asset)
Sort Order
0
Add Component

4 เขียน C# Script สำหรับการคำนวณ

- สร้างสคริปต์ CalculatorUI.cs และเพิ่มลงใน GameObject ที่มี UIDocument:


Create
Show in Explorer
Open
Delete
Rename
Copy Path Alt+Ctrl+C
Open Scene Additive
View in Package Manager
Import New Asset...
Import Package
Export Package...
Find References In Scene
Find References In Project
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Extract From Prefab
Update UXML Schema
Open C# Project
View in Import Activity Window
Properties... Alt+P
Folder
Material
MonoBehaviour Script
Prefab Variant
2D
Animation
Audio
Rendering
Scene
Scripting
Search
Shader
Shader Graph
Testing
Terrain
Text Core
TextMeshPro
Timeline
UI Toolkit
Visual Scripting
Physics Material
GUI Skin
Custom Font
Input Actions


Calculator
Calculator
CalculatorUI

CalculatorUI.cs

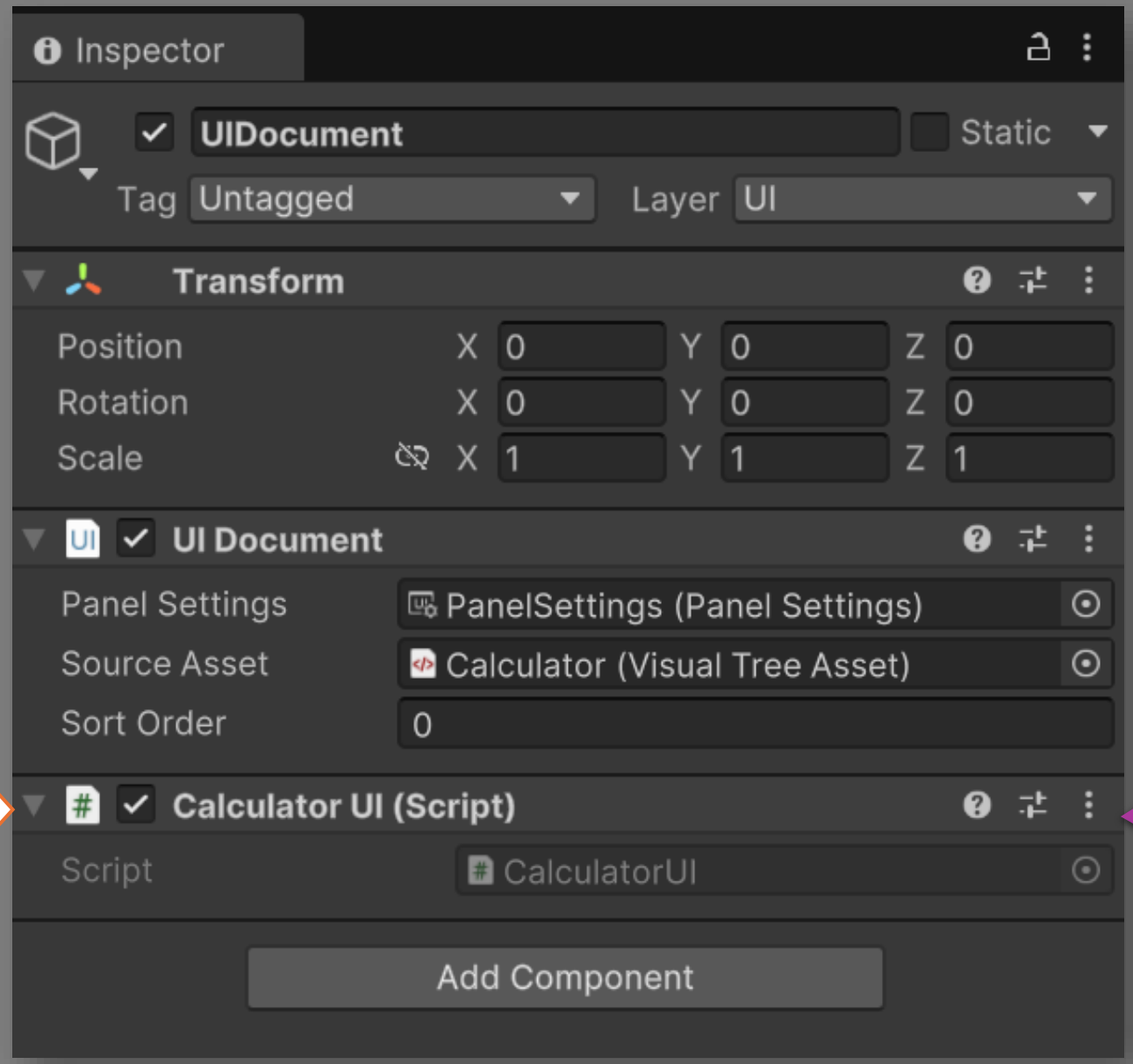

```
using UnityEngine;
using UnityEngine.UIElements;

public class CalculatorUI : MonoBehaviour
{
    private TextField input1;
    private TextField input2;
    private Button calculateButton;
    private Label resultLabel;
```

CalculatorUI.cs


```
void OnEnable()
{
    // ดึง UIDocument
    var uiDocument = GetComponent<UIDocument>();
    var root = uiDocument.rootVisualElement;
    // ค้นหาองค์ประกอบ UI
    input1 = root.Q<TextField>("input1");
    input2 = root.Q<TextField>("input2");
    calculateButton = root.Q<Button>("calculateButton");
    resultLabel = root.Q<Label>("resultLabel");
    // เพิ่ม Event ให้ปุ่ม
    calculateButton.clicked += CalculateSum;
}
```

CalculatorUI.cs


```
    private void CalculateSum()
    {
        // แปลงค่าจาก TextField เป็นตัวเลข
        if (float.TryParse(input1.value, out float num1) &&
            float.TryParse(input2.value, out float num2)) {
            float sum = num1 + num2;
            resultLabel.text = $"Result: {sum}";
        }
        else {
            resultLabel.text = "Invalid input. Please enter valid numbers.";
        }
    }
}
```



ลากโค้ดมาเชื่อมเข้ากับ UIDocument

ดึง UI องค์ประกอบ:

1. input1 และ input2 เป็นช่องรับตัวเลข
2. calculateButton คือปุ่มสำหรับคำนวณ
3. resultLabel คือฉลากสำหรับแสดงผลลัพธ์

เพิ่ม Event ให้ปุ่ม

เมื่อปุ่มถูกคลิก ฟังก์ชัน CalculateSum จะทำงาน

การคำนวณผลรวม:

1. ใช้ float.TryParse เพื่อแปลงค่าที่ผู้ใช้กรอกจาก TextField เป็นตัวเลข
2. หากค่าที่กรอกถูกต้อง จะทำการบวกเลขและแสดงผลใน resultLabel
3. หากไม่ถูกต้อง จะเตือนผู้ใช้ว่าอินพุตไม่ถูกต้อง

ตัวอย่าง

สร้าง Health Bar ของผู้เล่นด้วย UI Toolkit

ทำได้ง่ายโดยใช้ ProgressBar หรือ VisualElement ที่มีการกำหนดขนาดแบบปรับเปลี่ยนได้ตามค่าพลังชีวิต (Health)

1 สร้าง UI Layout ด้วย UXML

- สร้างไฟล์ HealthBar.uxml
- องค์ประกอบในไฟล์:
  - health-bar-container: โครงสร้างภายนอกสำหรับ
  - Health Barhealth-label: ป้ายข้อความที่แสดงคำว่า "Health“
  - health-bar: พื้นที่ที่กำหนดเป็นกรอบของแถบพลังชีวิต
  - health-fill: แถบภายในที่แสดงปริมาณพลังชีวิต (Health) ของผู้เล่น

HealthBar.uxml


```
<ui:UXML xmlns:ui="UnityEngine.UIElements">
    <ui:VisualElement class="health-bar-container">
        <ui:Label text="Health" class="health-label" />
        <ui:VisualElement class="health-bar">
            <ui:VisualElement class="health-fill" name="healthFill" />
        </ui:VisualElement>
    </ui:VisualElement>
</ui:UXML>
```

2 กำหนดสไตล์ด้วย USS

- สร้างไฟล์ HealthBar.uss
- สไตล์ที่กำหนด:
  - health-bar: เป็นกรอบที่มีสีพื้นหลังเข้ม
  - health-fill: สีเขียวที่ปรับความกว้างตามค่า Health ของผู้เล่น

HealthBar.uss


```
.health-bar-container {
    display: flex;
    flex-direction: column;
    align-items: flex-start;
    width: 300px;
    padding: 10px;
    background-color: #333;
    border: 2px solid #000;
    border-radius: 8px;
}
```

HealthBar.uss


```
.health-label {
    font-size: 18px;
    color: white;
    margin-bottom: 5px;
}
.health-bar {
    width: 100%;
    height: 20px;
    background-color: #555;
    border-radius: 5px;
    overflow: hidden;
}
```

HealthBar.uss


```
.health-fill
    width: 100%; /* จะปรับตามค่า Health */
    height: 100%;
    background-color: #4caf50; /* สีเขียว */
   transition: width 0.3s; /* เพิ่มอนิเมชัน */
}
```

3 เชื่อมต่อ UXML และ USS กับ UIDocument

- สร้าง GameObject
  - ใน Unity Scene ให้สร้าง GameObject และเพิ่มคอมโพเนนต์ UIDocument
- เชื่อม UXML
  - ใน Inspector ของ UIDocument ให้ตั้งค่า Source Asset เป็นไฟล์ HealthBar.uxml
- เชื่อม USS
  - ในไฟล์ UXML จะมีการอ้างอิงไฟล์ HealthBar.uss โดยอัตโนมัติหากสร้างผ่าน UI Builder

4 Script สำหรับควบคุม Health Bar

- สร้างไฟล์ HealthBarController.cs
- ฟังก์ชันใน Script:
  - OnEnable ดึงไฟล์ UXML และกำหนดค่าเริ่มต้นให้กับ Health Bar
  - TakeDamage ลดค่า currentHealth เมื่อผู้เล่นได้รับความเสียหาย และอัปเดตแถบพลังชีวิต
  - UpdateHealthBar คำนวณความกว้างของแถบ health-fill ตามเปอร์เซ็นต์พลังชีวิตที่เหลือ

HealthBarController.cs


```
using UnityEngine;
using UnityEngine.UIElements;

public class HealthBarController : MonoBehaviour
{
    private VisualElement healthFill;
    private float maxHealth = 100f;
    private float currentHealth;
```

HealthBarController.cs


```
void OnEnable()
{
    // ดึง UIDocument
    var uiDocument = GetComponent<UIDocument>();
    var root = uiDocument.rootVisualElement;
    // ค้นหาองค์ประกอบ Health Fill
    healthFill = root.Q<VisualElement>("healthFill");
    // ตั้งค่าเริ่มต้น
    currentHealth = maxHealth;
    UpdateHealthBar();
}
```

HealthBarController.cs


```
    public void TakeDamage(float damage)
    {
        currentHealth -= damage;
        // จำกัดค่าให้อยู่ระหว่าง 0 ถึง maxHealth
        currentHealth = Mathf.Clamp(currentHealth, 0, maxHealth);
        UpdateHealthBar();
    }
    private void UpdateHealthBar()
    {         // คำนวณความกว้างของแถบพลังชีวิต
        float healthPercentage = currentHealth / maxHealth;
        healthFill.style.width = new Length(healthPercentage * 100,
                                            LengthUnit.Percent);
    }
}
```

5 ทดสอบใน Scene

- เพิ่มคอมโพเนนต์ HealthBarController ลงใน GameObject เดียวกันกับ UIDocument
- สร้างฟังก์ชันใน Game Manager หรือ Object อื่นที่เรียก TakeDamage() เพื่อจำลองการลดพลังชีวิต
- กด Play เพื่อทดสอบ:เมื่อกดปุ่ม Space แถบพลังชีวิตจะลดลงตามค่าที่กำหนด

ตัวอย่างการเรียกใช้


```
void Update()
{
    if (Input.GetKeyDown(KeyCode.Space)) // กด Space เพื่อลดพลังชีวิต
    {
        healthBarController.TakeDamage(10f);
    }
}
```

เมื่อผู้เล่นได้รับความเสียหาย เช่น จากศัตรูหรือเหตุการณ์ในเกม ตัวฟังก์ชัน TakeDamage จะถูกเรียกใช้เพื่อลดค่าพลังชีวิต และแถบ health-fill จะปรับความกว้างเพื่อแสดงผลลัพธ์ที่สอดคล้องกับค่าพลังชีวิตที่เหลืออยู่

Q&A