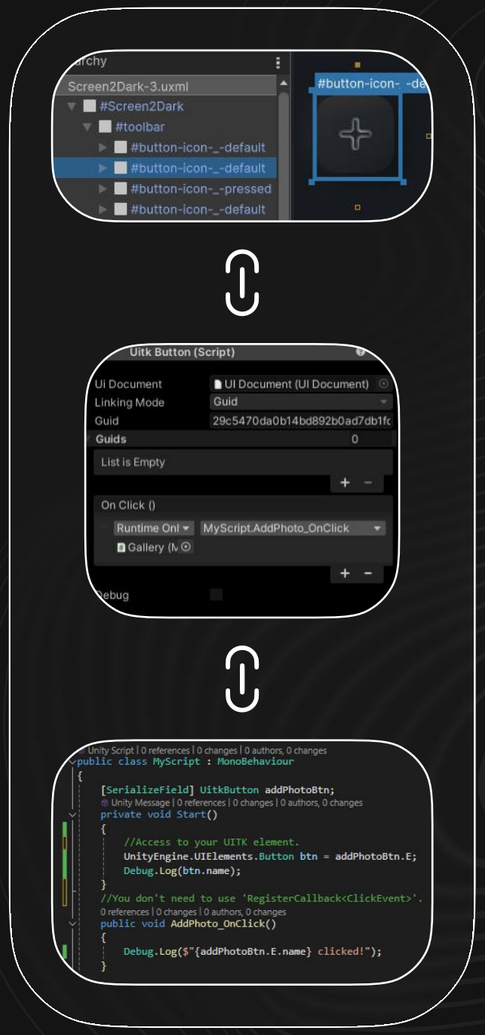


UITK Element Linker

manual for developers

0.0.1

Introduction

I strongly recommend reading
this guide before using the asset.

1. This asset is created to simplify the development of projects that use UI Toolkit.
The asset is completely free and distributed under the MIT license.
You can contribute to the project on GitHub and suggest improvements to the asset.

2. The list of supported elements is indicated in the description of the asset (in the GitHub repository, or on the Asset Store page).

3. If you encounter any errors while working with the asset, please write me about it at provided contacts or in the Issues section on GitHub.
I typically respond quickly to messages, offer assistance on an individual basis, and address any identified errors in the upcoming updates.

Discord Server: https://discord.com/invite/ZsnDffV5eE

Telegram Group: -

Telegram Support: https://t.me/da_assets

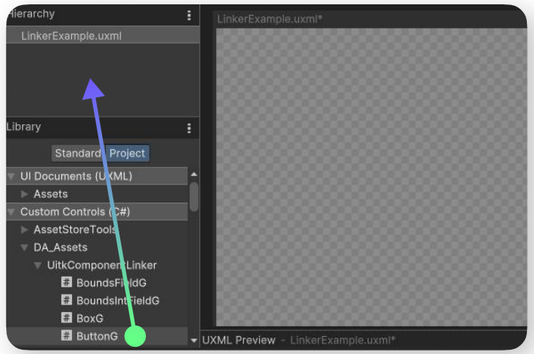
Email Support: da.assets.publisher@gmail.com

Website: https://da-assets.github.io/site/

GUID Linking

1. To set up linking by **Guid**, open your UXML layout, go to the **Library** section, and click on the **Project** tab.

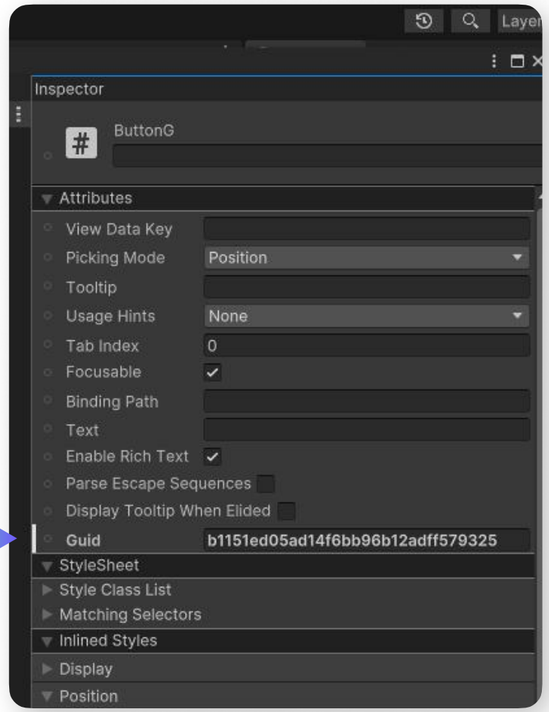
   In the list of elements, find the "**UitkElementLinker**" asset elements, and drag the necessary element with the "**G**" prefix into your layout.



2. 

   Click on the added element to open its inspector.

   In the inspector, you will see the standard fields of the element from which the **G-element** is inherited (in this case, a button), as well as a new field "**Guid**".



   > ***The value in the Guid field should be in "override" state**, meaning it should have a **white stripe** next to the field.*
   > *If there is no white stripe, it means that upon reopening the UXML file, the guid in the field will change to a random one, and the **linking will stop working.***

   To switch the field to "override" state, you can manually add any character to the current field value and then remove it.

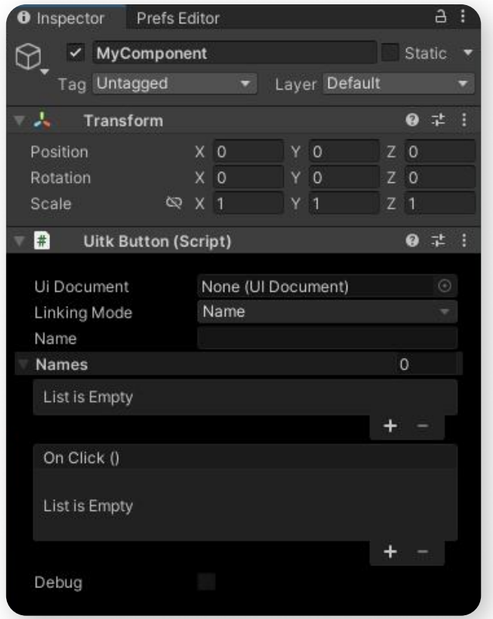
   After this, be sure to **save your UXML file.**

GUID Linking

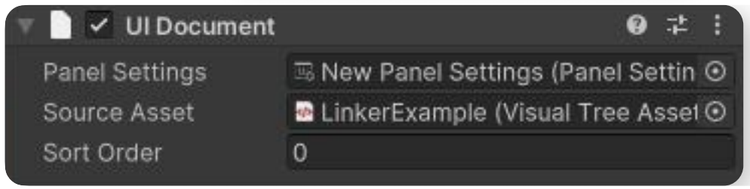
3 After you have set up the **Guid**, create an empty GameObject on the scene and add a component that will facilitate the linking between your UITK element and the MonoBehaviour script.

The name of the linking script is formed as follows:
"**Uitk**" + **ElementName**.

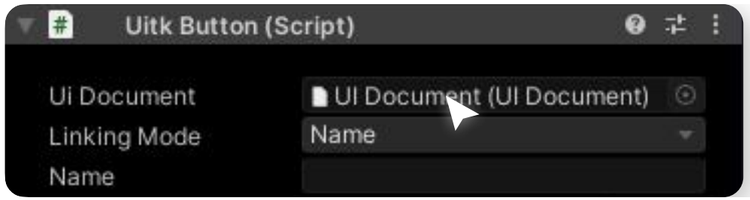
In our case, the script is called "**UitkButton**".



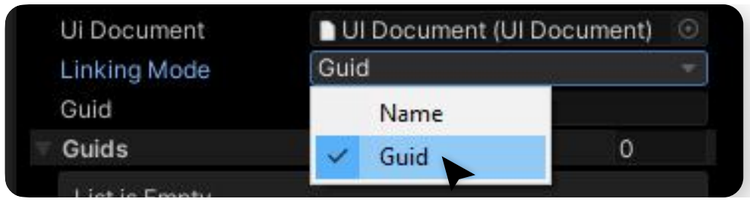
2 If you don't have a **UIDocument** component, create an empty **GameObject** on the scene, add the **UIDocument** component to it, serialize your UXML file in it, as well as the **PanelSettings** file.
You can learn more about **PanelSettings** here.



3 Drag the GameObject, to which the UIDocument script has been added, into the "**Ui Document**" field of the UitkButton script.

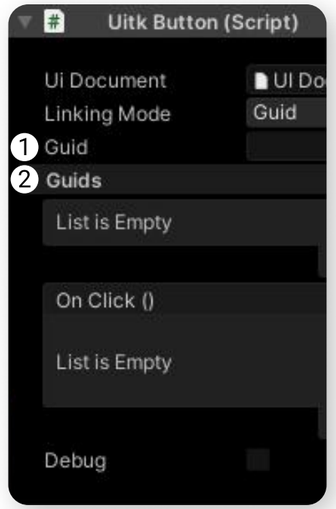


4 Switch the "**Linking Mode**" parameter to the "**Guid**" state.



GUID Linking

5 Now you can choose one of **two possible methods** to link the element via GUID.



1. Using the "**Guid**" field, linking with a **single guid**.
In this case, the element search will be conducted throughout the hierarchy of your layout.

This method is **not** very **efficient**, especially **for large and complex layouts**, but it is convenient if the upper hierarchy of your element often changes during development.

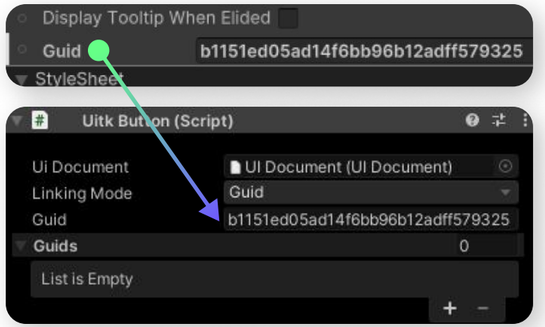
When searching for an element using the "**guid**" field, the parents of the sought element can be both **G-elements** and regular elements from the UnityEngine.UIElements namespace.

2. Using the "**Guids**" field, linking with a **hierarchy** of the element.
That is, the GUIDs of all the parents of your element will be used to search for the element. In this case, the element search will only be conducted along one branch of the hierarchy, ignoring other parts of the hierarchy where your element is not present.

This is the **most efficient way** to find a element, but at the same time, if you change the parent of your element, or any other parent in the upper hierarchy of your element - you will need to update the "**Guids**" field for your element, otherwise it will not be found.

As a suggestion, use linking via the "**Guid**" field during development/prototyping, and switch to "**Guids**" before release when you will no longer be making changes to your layout, to improve the performance of your game/app.
In such a case, all the elements of the hierarchy branch where your element is located must be **G-elements**.

6 To link your element using the "**Guid**" field, following the first method, insert the **guid** of your element into the "**Guid**" field.



Keep in mind that if there are more than 0 elements in the "**Guids**" array, linking through the "**Guid**" field will not work, as the asset will assume that you are linking through the "**Guids**" field.

GUID Linking

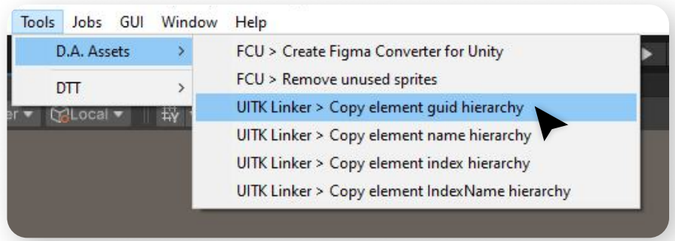
7

To link your element through the "**Guids**" field, following the second method:

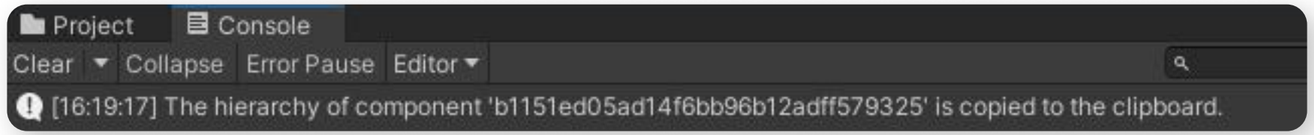
1. Open your UXML layout

2. Select your element by clicking it once with the left mouse button

3. In the context menu at the top of the Unity window (not in the UI Builder window), click on "**Tools** > **Assets** > **Copy element guid hierarchy**" item.

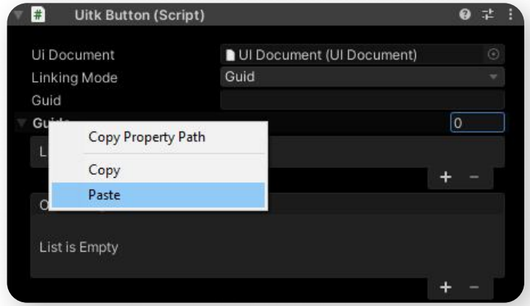


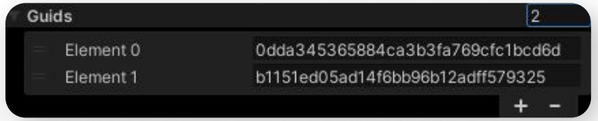
4. If the hierarchy was copied successfully, you will see a message in the console:



5. If you see an error when trying to copy a hierarchy, please notify the asset developer.

6. Now you have configured element linking using the "**Guids**" field.

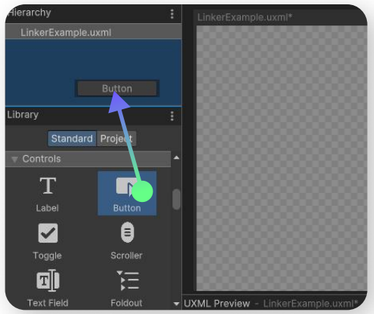


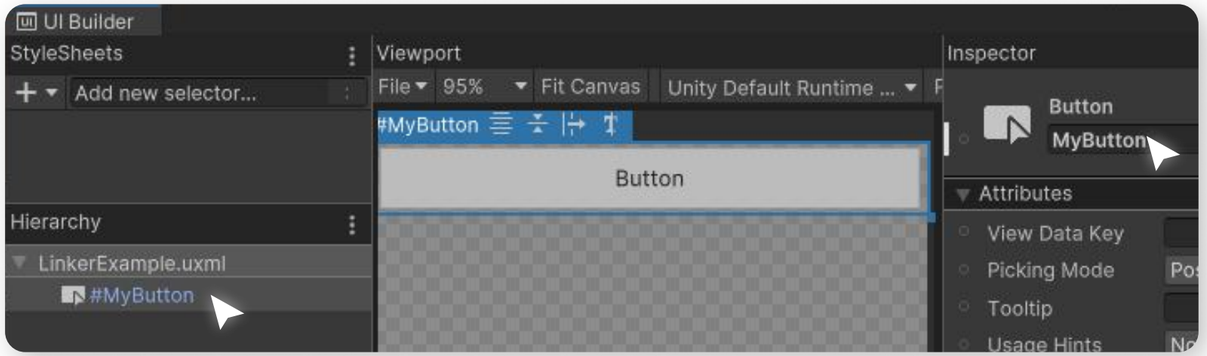


Name Linking

1. Now let's consider linking by Name and/or Index.
Linking by name works with both regular UI Toolkit elements and G-elements.

Open your UXML layout, go to the **Library** section, and click on the **Standard** tab.
Drag any of standard elements onto your layout in the UI Builder window.



2. 

Click on the added element to open its inspector.

Then assign a name to it and copy this name to the clipboard.

After this, be sure to **save your UXML file.**

Name Linking

3 Open your UitkButton component, which is on the GameObject in the scene, in the Inspector, and switch the "**Linking Mode**" to "**Name**."

1. If you want to configure the binding by name only, enter the name of your element in the "**Name**" field.

In this case, the search for the element will be conducted across all branches of your layout's hierarchy, and linking will be established with the first element whose name matches the value in the "**Name**" field.

This approach uses the standard "**Query**" method to search for an element in the hierarchy, and is not recommended for use because if your layout contains repeated names, there is a high risk of linking with the wrong element.

Keep in mind that if there are more than 0 elements in the "**Names**" array, linking through the "**Name**" field will not work, as the asset will assume that you are linking through the "**Names**" field.

Name Linking

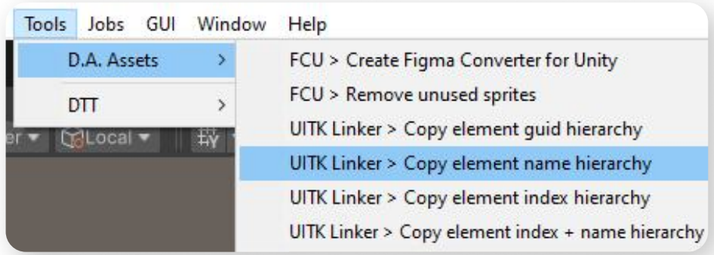
2. The problem described above can be solved by linking through the "**Names**" field.
Add the first element to the "**Names**" array using the "**+**" button in the Inspector.

You will see two fields in the resulting array element: "**Index**" and "**Name**".
"**Index**" is the ordinal number of your element in the hierarchy within its parent relative to other elements.
"**Name**" is the name of your element.

By default, the Index field has a value of "**-1**", which means that index checking at this level of hierarchy is not used, and the object will be searched only by "**Name**".

However, if an "**Index**" is specified, the asset will search for your element by index, not by "**Name**", even if you have specified a "**Name**".

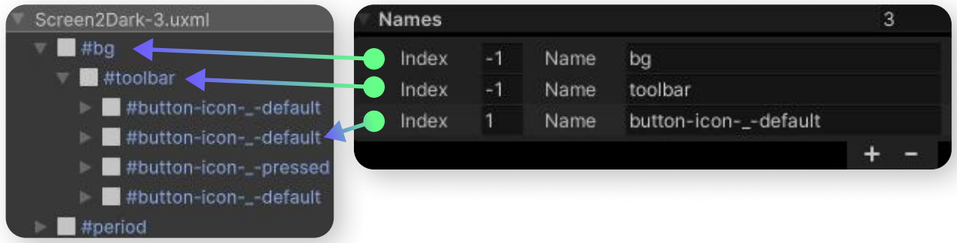
To copy the hierarchy of Names and/or indexes, use the context menu items:



- fill the hierarchy with "**Name**" field only
- fill the hierarchy with "**Index**" field only
- fill the hierarchy with both field “**Index**” and field "**Name**"

Name Linking

3. Below is an example of using the "**Names**" linking field, where I used the copy function for "**Index + name**". However, for elements that do not have duplicates at their hierarchy level, I changed the indexes to "**-1**" so they would not be considered during the search.
For the element with the repeated name "button-icon-_-default", index "**1**" is specified - in this case, we want to get the second button in the hierarchy.



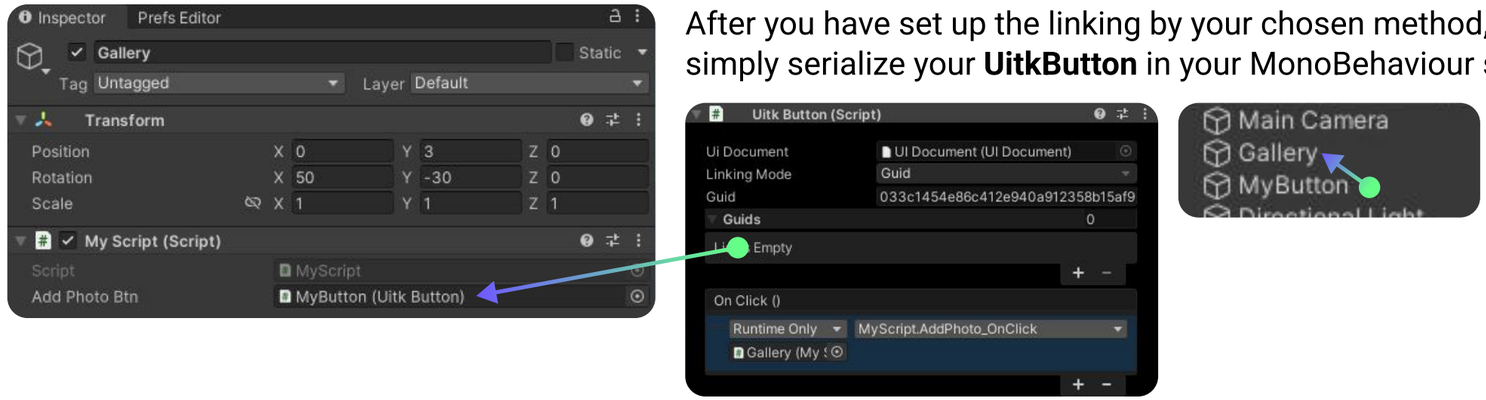
4. Disadvantages of using the Names field for binding:

a) The indexes of your components may change during development or at runtime.
b) The names of your components may change during development.

I recommend using the "**Guid**" or "**Guids**" field for linking to avoid the problems described above.

Access to the element

1



After you have set up the linking by your chosen method, you can simply serialize your **UitkButton** in your MonoBehaviour script.

2

To access the UnityEngine.UIElements.**Button** element, to which you've configured the linking, you can refer to the properties of the **UitkButton** component: **"E"**, or "**Element**".

Example code:

```
using DA_Assets.UEL;
using UnityEngine;

public class MyScript : MonoBehaviour
{
    [SerializeField] UitkButton addPhotoBtn;
    private void Start()
    {
        //Access to your UITK element.
        UnityEngine.UIElements.Button btn = addPhotoBtn.E;
        Debug.Log(btn.name);
    }
    //You don't need to use 'RegisterCallback<ClickEvent>'.
    public void AddPhoto_OnClick()
    {
        Debug.Log($"{addPhotoBtn.E.name} clicked!");
    }
}
```