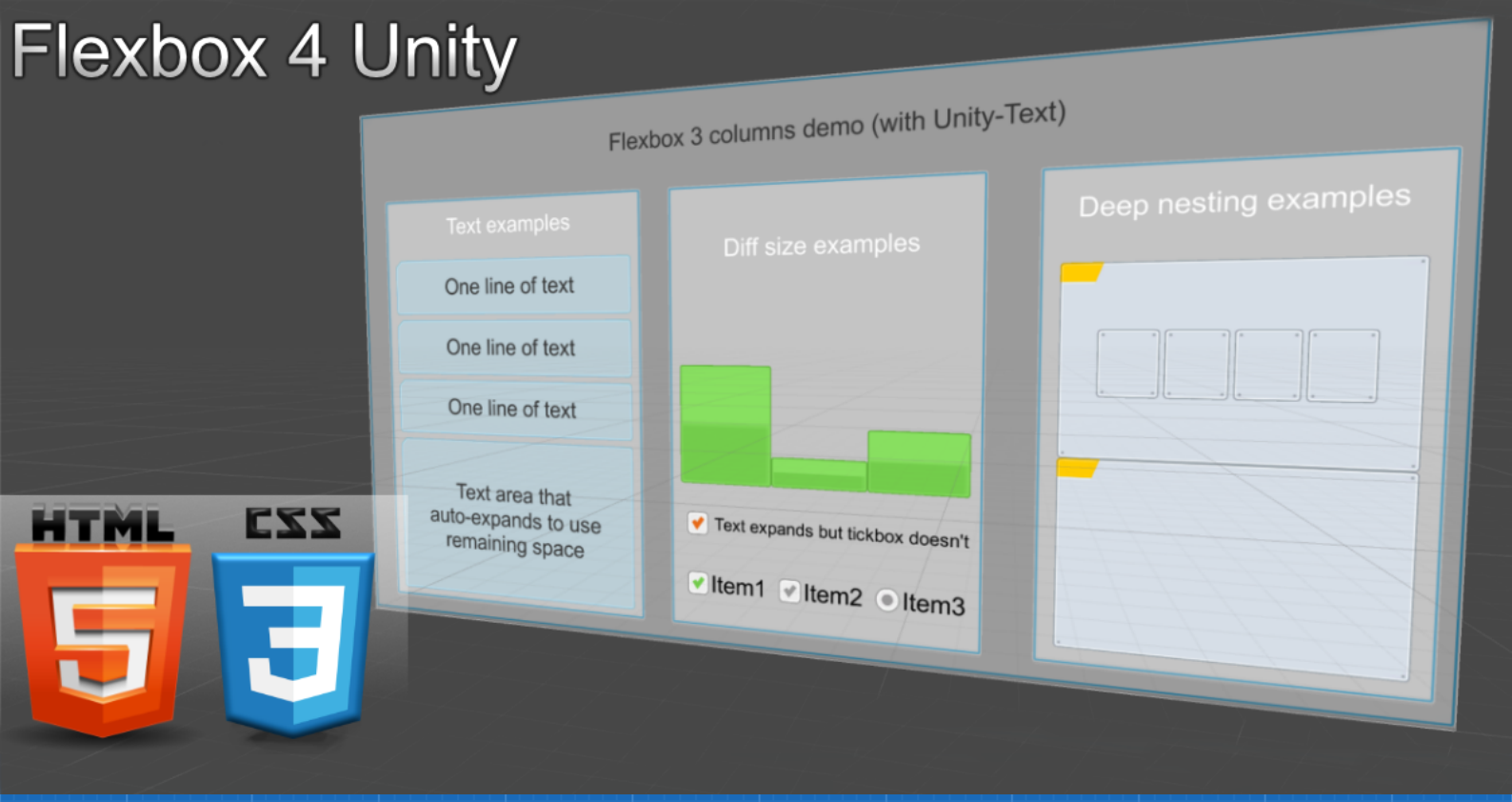


Flexbox 4 Unity v2.4

(March 2020)

Good choice! Flexbox will make you faster at building GUI/UI, and give you more responsive, adaptive designs that flow easily and layout more cleanly.

Table of Contents

Quick Start....2
1. Create a flex-container....2
2. Create a flex-item....3
3. Add a Text component to the flex-item....3
4. Edit the text to say something:....4
5. Select the FlexItem in hierarchy and duplicate it:....4
6. Using the live-preview system....5
Main features / Special Techniques....7
Embedding FlexContainers inside Unity's default GUI / RectTransforms....7
Embedding UnityUI inside FlexContainers / FlexItems....7
ASPECT_FIT: fixed-aspect GUIs inside variable-aspect GUIs....8
Using Padding and Margins to create beautiful GUIs....9
Padding vs Margins....9
Using Margin with Flexbox4Unity....9
Using Padding with Flexbox4Unity....10
Flexbox changes vs CSS3....11
Removed/Not implemented: CSS-3 Anonymous FlexItems....11
Percentages are applied to parent width OR height....11
Not supported yet: flex-wrap....11
Using Flexbox for rich, complex, GUIs....12
Online guides....12
Quick overview....12
Help!....12
Troubleshooting....13
FAQ: Support forum....13

FAQ: My GUI scales weirdly when I change resolution....13
FAQ: “NullReferenceException: Object reference not set to an instance of an object”....13
Recent changes....14
2.4: Improvements to VR support, new Experimental flex-wrap....14
2.3: Complete re-write of the Layout engine (fixed many bugs + edge-cases)....14
2.2: Improved editing GUI....14
2.1: New layout-algorithms....14
2.0: Big improvements for Unity 2018, Unity 2019....14
1.5: Replaced UnityUI's slow algorithm with a custom fast one....14
1.4: Added “padding” and improved auto-sized content....14
1.3: Performance optimizations....14
1.2: Greatly improved “margins” support....14
1.1: Fix Unity prefabs error, improve Aspect-Ratio Fill....14
1.0: First public release....14

Quick Start

Flexbox4unity implements the HTML5/CSS3 layout system called “Flexbox”. There are two parts to Flexbox layouts:

- flex-container
- flex-item

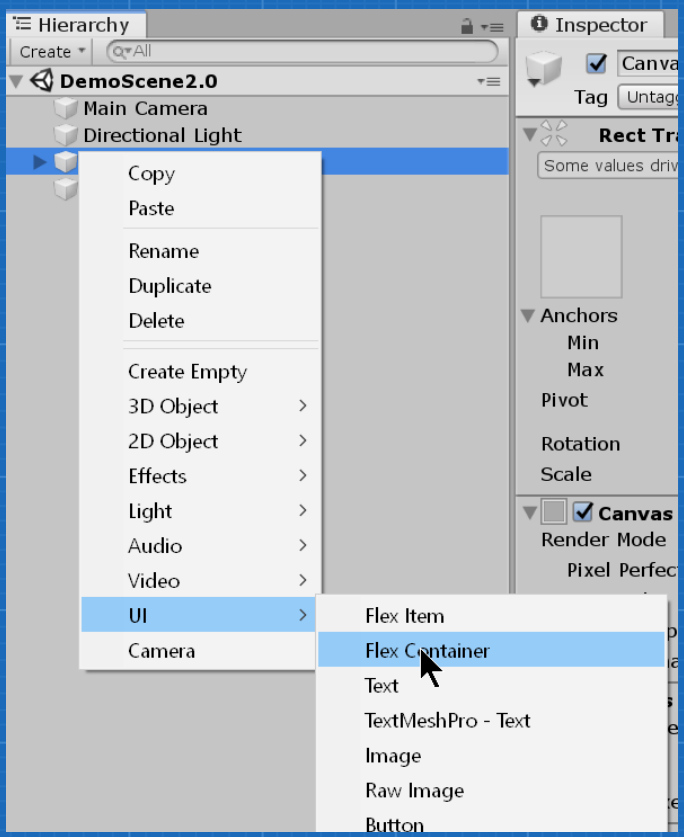
To create a Flexbox layout, you need at least one flex-container, and at least one flex-item inside it. In Unity, these are represented by separate Components / MonoBehaviours:

- FlexboxLayoutGroup = a Unity3D LayoutGroup that implements flex-container
- FlexboxElement = a Unity3D LayoutElement that implements flex-item

1. Create a flex-container

Add a Canvas to your scene if it doesn't already have one, then right-click it and select:

UI > Flex Container

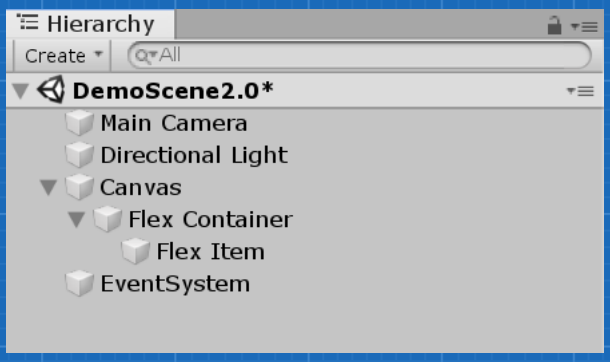


2. Create a flex-item

Select the new flex-container in the Hierarchy window, and add a Flex item using:

UI > Flex Item

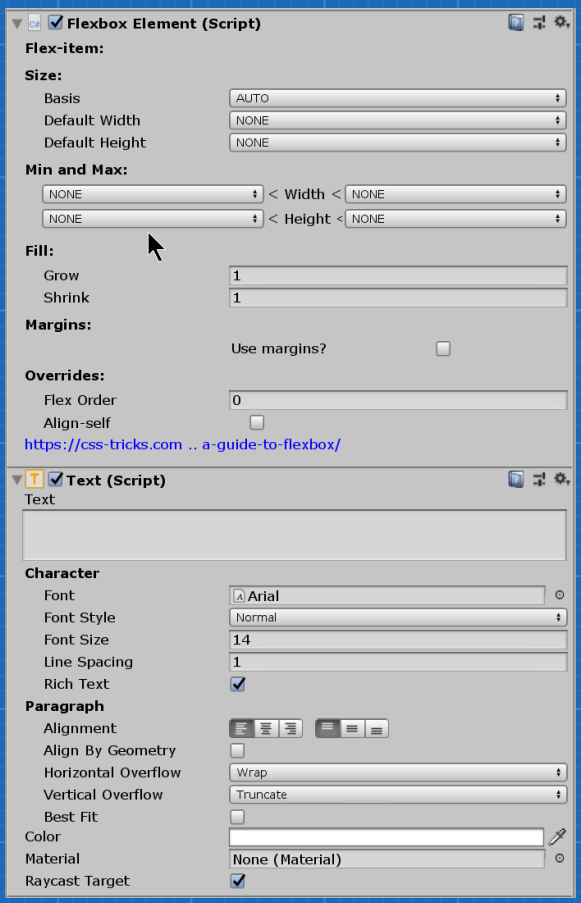
You should now have the flex-container and flex-item on your Canvas like this:



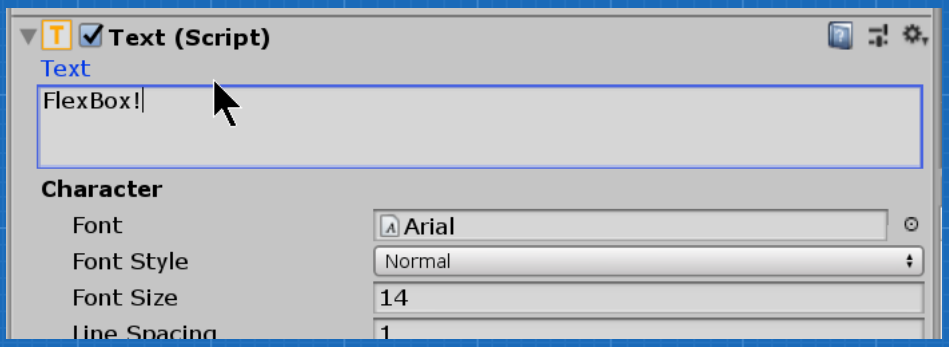
3. Add a Text component to the flex-item

Flexbox4Unity is fully integrated with Unity's built-in GUI layout system, so you can use any components from core Unity directly in Flexbox.

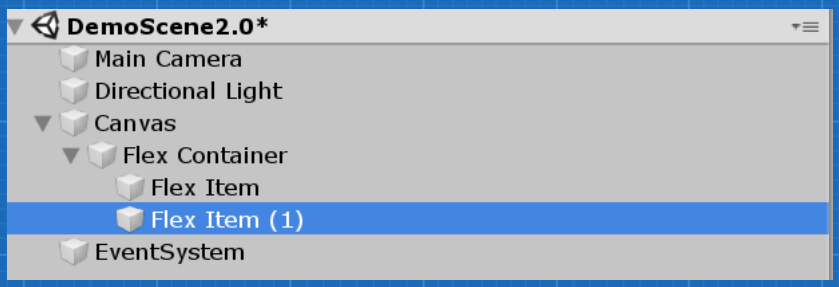
First, add a Text component to the flex item:



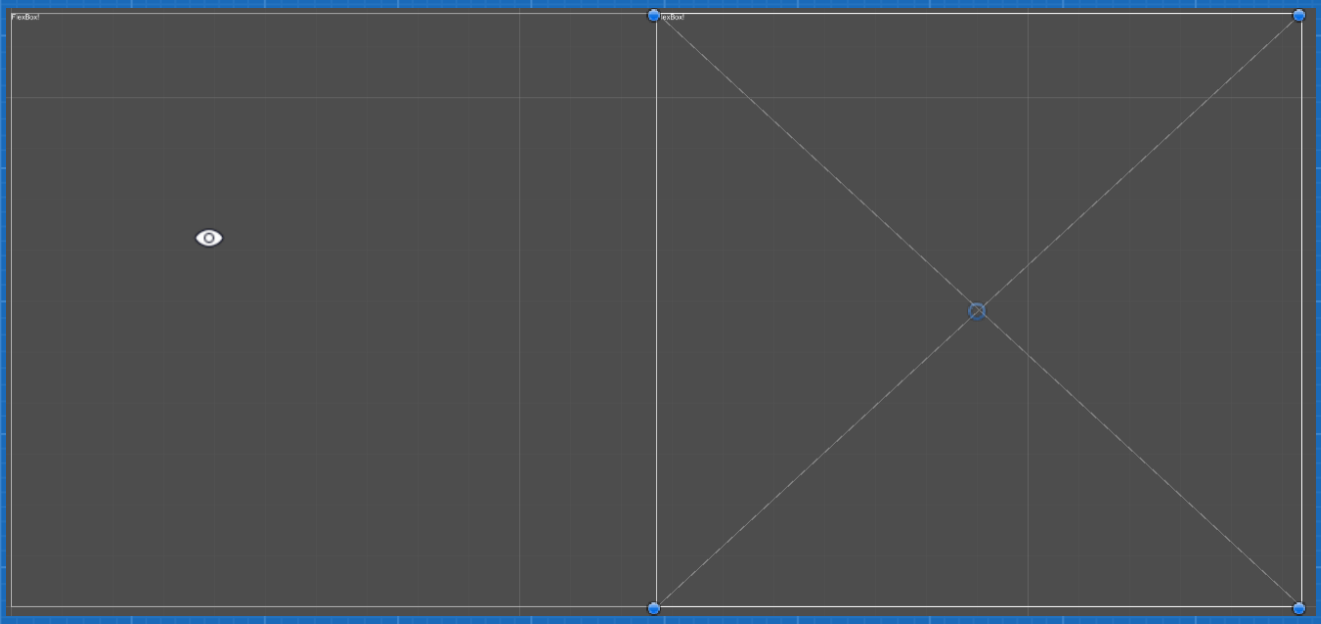
4. Edit the text to say something:



5. Select the FlexItem in hierarchy and duplicate it:



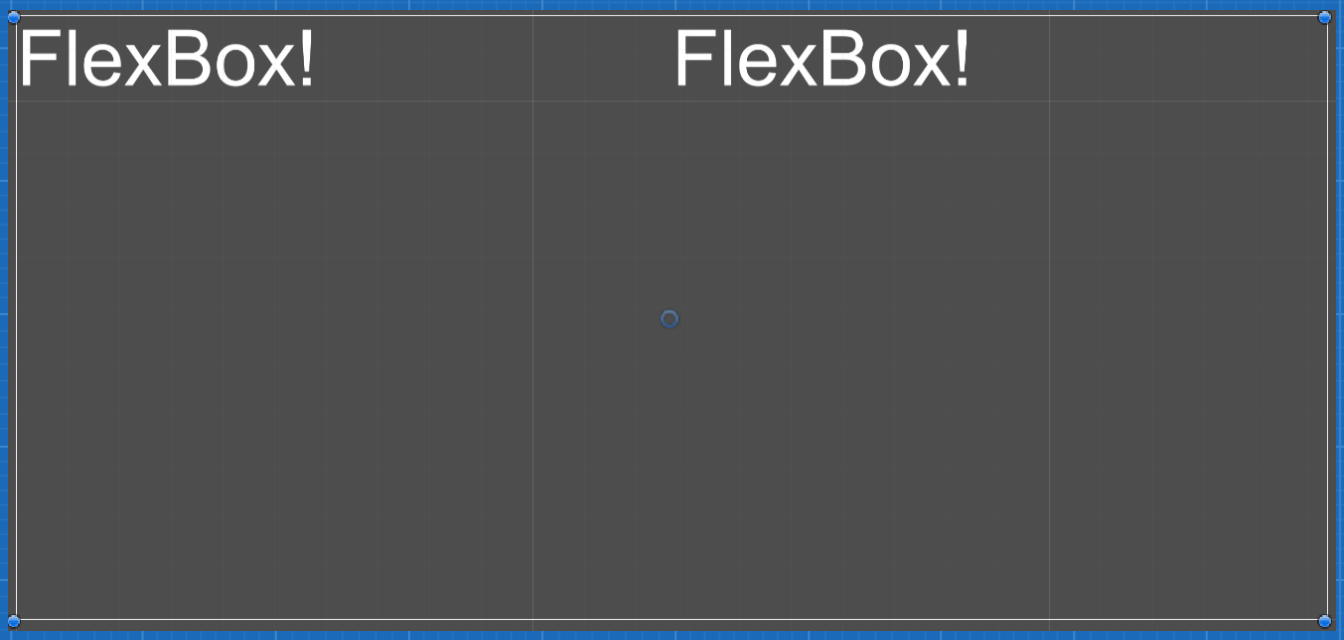
Flexbox4Unity will now automatically layout your two text items, and it should look something like this in the SceneView:



Edit the two Text components, and set:

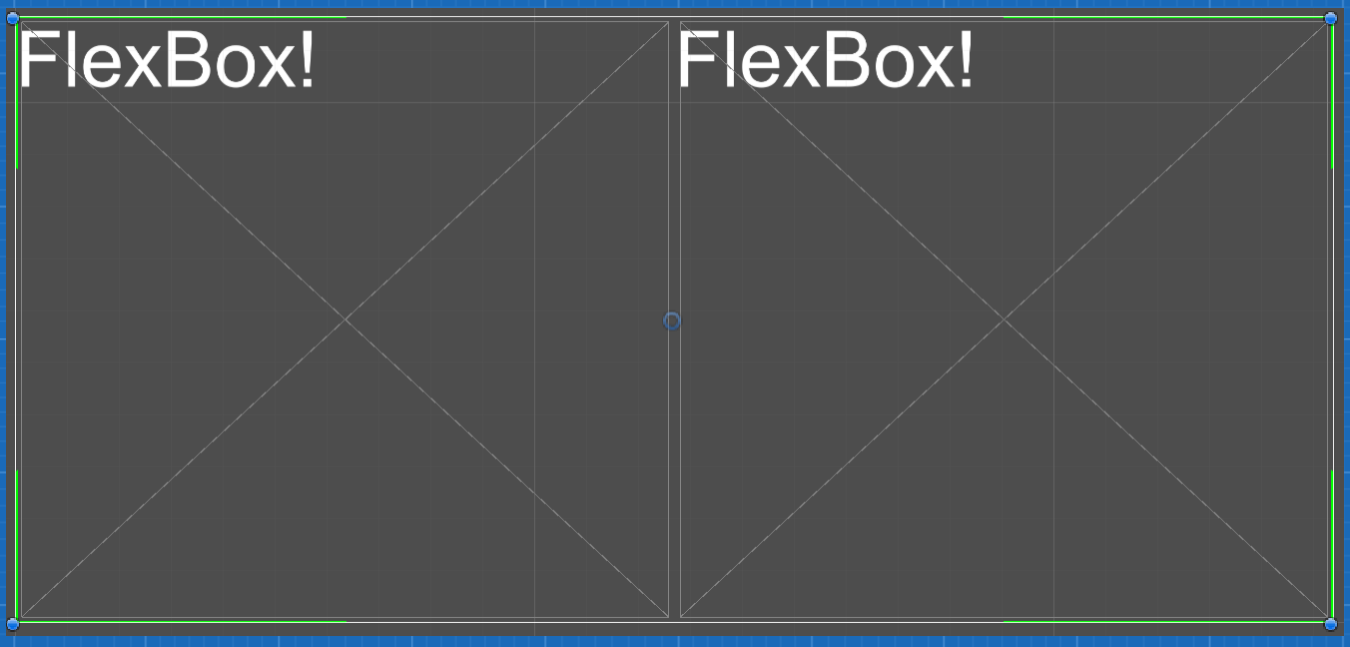
1. Color = white
2. Font-size = 150

...and your SceneView should now look like this:



6. Using the live-preview system

In the Hierarchy view, try selecting the main Flex Container you created. This will show the live-preview of your Flexbox hierarchy, making it easier to rapidly edit and debug complex layouts:

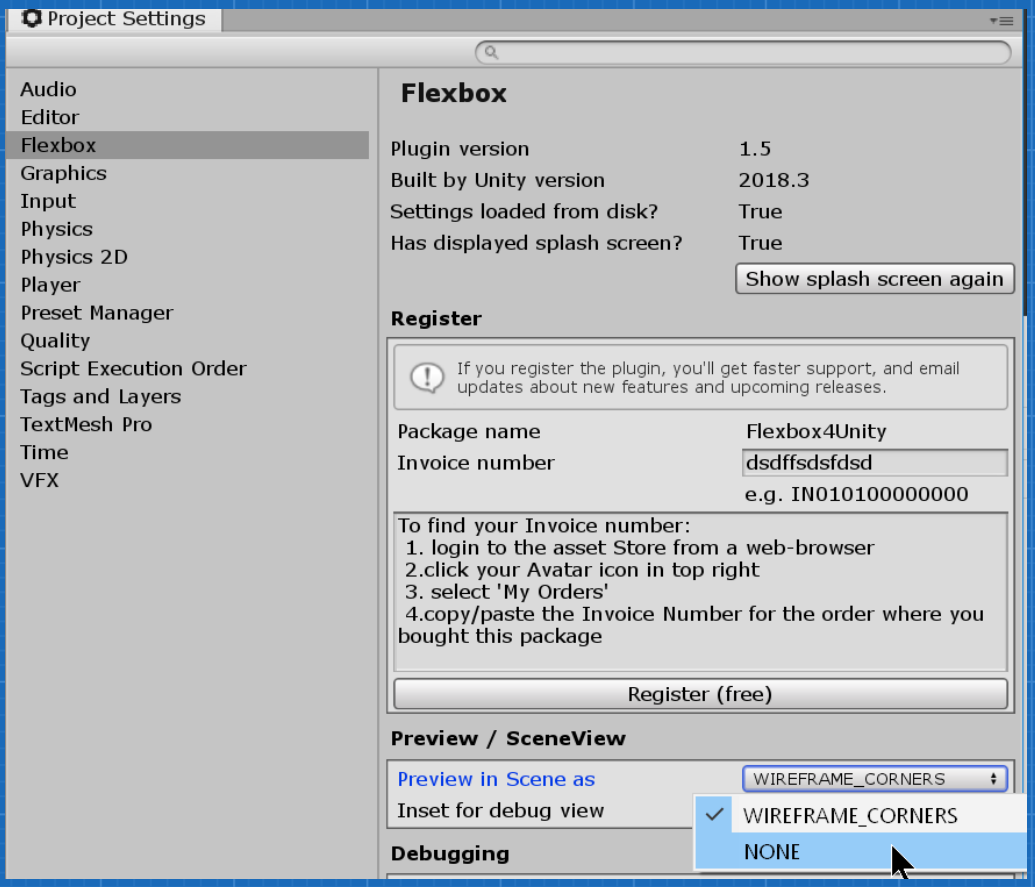


The grey crossing lines show where child FlexItem objects are.

The green corner lines show where nested FlexContainer objects are (try adding child FlexContainers, and you'll see additional sets of green-corners appear).

Note: you can disable preview mode at any time by going to your Unity Project Settings:

Edit > Project Settings > Flexbox



Embedding FlexContainers inside Unity's default GUI / RectTransforms

Use Unity's anchors and handles to position and control the flex-container however you want. e.g. using the RectTransform tool put the anchors in the four corners of screen, and drag the blue dots to the four corners, making it auto-size to fit your screen.

Unity will control the size and position of the parent container, and Flexbox4Unity will control layout inside the container.

Embedding UnityUI inside FlexContainers / FlexItems

You can use any default UnityUI control within Flexbox4Unity, and it will be automatically resized by Flexbox.

There are two main ways of doing this, with pros/cons of each. Both require a FlexItem component to control the layout.

Option 1: Attach a FlexItem component to a UnityUI GameObject

1. Select the gameobject with the FlexItem
2. (recommended): Set the flex-basis to "CONTENT"
3. (recommended): Set the flex-grow to "0"

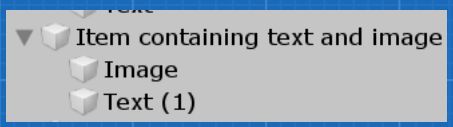
...the FlexItem will now intelligently resize itself to the best size for the UnityUI element (e.g. a Unity.UI.Button will try to wrap the text inside it, but will allow the text to flow to multiple lines if necessary).

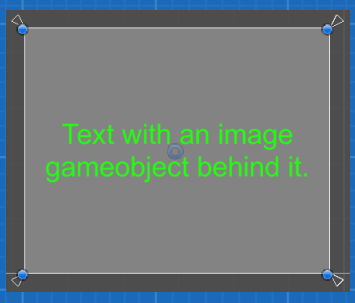
Option 2: Parent multiple UnityUI GameObjects inside a FlexItem GameObject

1. Select the gameobject with the FlexItem
2. (recommended): Set the flex-basis to "CONTENT"
3. (recommended): Set the flex-grow to "0"
4. Create a child object for each UnityUI item
5. Select each child object and drag the anchors + handles to auto-fill the parent

…in this case, the FlexItem will choose the most important UnityUI object among its children, and use that as the basis of the size. The order of importance is currently: Button > Toggle > Inputfield > Text > (others).

e.g. here is a Unity Image background with a Unity Text on top, both of them resized to always stay together, but automatically resizing and fitting the text:





ASPECT_FIT: fixed-aspect GUIs inside variable-aspect GUIs

The CSS-3 specification of FlexBox was designed for web browsers, and does not automatically adapt to different sized/shaped screens (CSS has a different mechanism for that – @screen – which you could implement in Unity but is much more complex to use).

The only major missing feature that we need in Games/Apps is control of the AspectRatio of parts of the GUI. We want this for mobile games, desktop games, embedded RenderTextures, etc. I have extended the CSS-3 Flexbox specification to include an extra value of flex-basis: "ASPECT_FIT".

To use ASPECT_FIT:

1. Create a FlexContainer (or choose an existing one)
2. Add a child object, with a FlexItem component
3. Set the child's flexBasis to "ASPECT_FIT"
4. In the field for flexBasis, type the aspect ratio as X/Y.
    1. e.g. to force 1:1 ratio, type "1"
    2. e.g. to force 3:2 ratio, type "1.5"
    3. e.g. to force 3:4 ratio, type "0.75"
5. The FlexItem will now always maintain that Aspect ratio, and will fit itself as large as possible within the parent FlexContainer.

Demo Scene: "**DemoScene – imacs**"

Using Padding and Margins to create beautiful GUIs

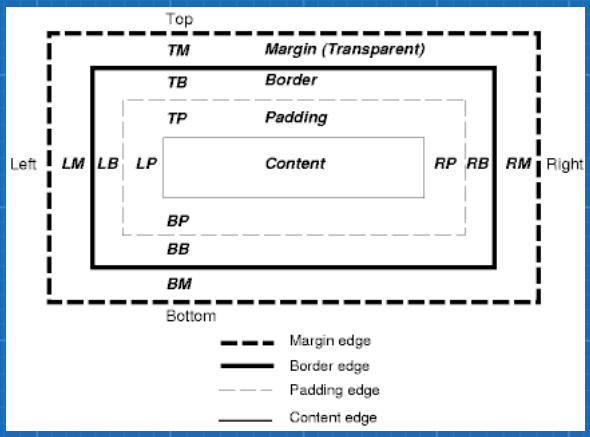
The power of Flexbox is its simplicity. The flex system makes it very easy for us to control the position and size of everything in our GUI (and in most cases we can leave Flexbox at its default settings, where it intelligently does the the layout for us).

But to get really good-looking GUIs, you need to control the empty space as well: the padding, the margins, the indentation.

CSS-3's Flexbox is tightly integrated with CSS's margins/padding system, so I have also implemented that part of CSS – you cannot fully run Flexbox without giving it detailed information about the margins/padding of every item (and Unity does not support margins/padding itself, so we have to add it).

Padding vs Margins

Here is the official image from CSS-3 showing how they relate to each other:



To recap:

- Padding – extra space INSIDE the UI element, but AROUND/OUTSIDE its main content
- Margin – extra space OUTSIDE the UI element

The main differences in practice:

- If a UI element has text – e.g. a button, with text – then Padding increases the space around the text (but inside the button image), while Margin increases the space outside the button (transparent background).
- The background of an element covers the Padding but doesn't cover the Margin

Using Margin with Flexbox4Unity

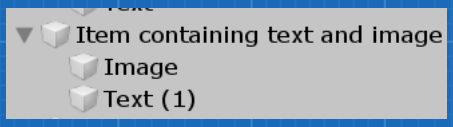
Margins **can be applied by any FlexItem**, and affect any attached UI elements / RectTransforms, and any children.

Margins tell the parent FlexContainer to put empty space around the FlexItem and its children.

Using Padding with Flexbox4Unity

Padding **can only be applied by a FlexItem that is also a FlexContainer**, and only affects the child UI elements / RectTransforms.

The classic setup is: a GameObject with FlexItem, and a child GameObject with Text/Button/etc, with the child RectTransform expanded to fit the full size of the parent. Each child also needs a FlexItem component (so that the parent FlexContainer can control its size), but you usually leave these at default settings and/or set them to “basis = CONTENT” (or basis = AUTO).

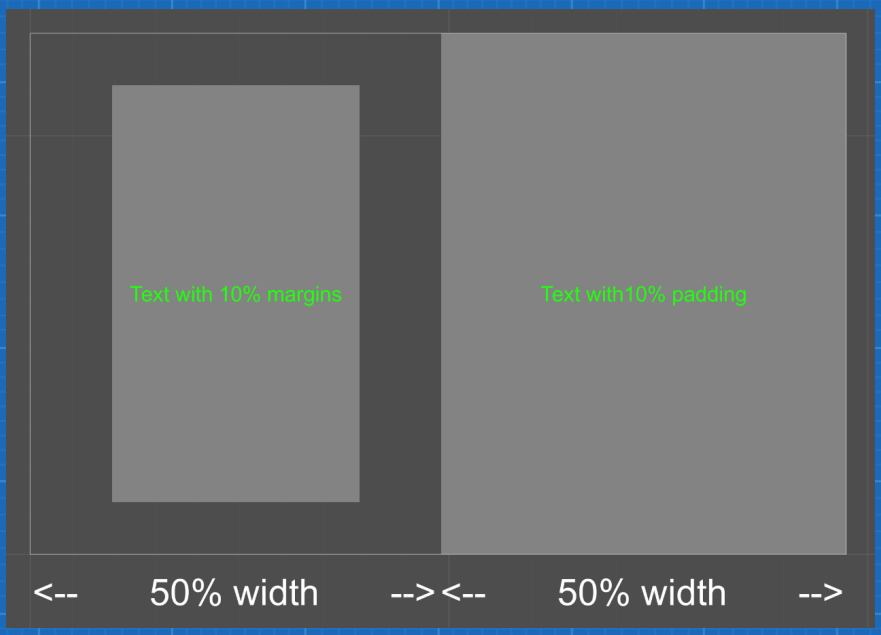


Note: if you attach the FlexItem directly to a UI element (e.g. a Button or an InputField), it will be ignored – this is because Unity’s UI system doesn’t support Padding internally, and only Flexbox4Unity understands how to use it. The UnityUI built-in components cannot handle padding on their own.

(this is why you need to make the item with padding both a FlexItem and a FlexContainer – it allows Flexbox4Unity to implement the missing features from UnityUI).

DemoScene: “margins and padding”:

This scene contains a simple example of two identical FlexItems, but one using margins, the other using padding.



Flexbox changes vs CSS3

Removed/Not implemented: CSS-3 Anonymous FlexItems

CSS's "Anonymous FlexItems" make it impossible to fully integrate with Unity's UI / RectTransform / other UI libraries and plugins.

(In CSS, every child of a FlexContainer is automatically a FlexItem – you cannot have non-flexitems as children)

In CSS, the workaround is to use other CSS systems – e.g. “position: absolute” – to “un-flex” the automatically-flexed items; to keep things simple in Unity, I have instead removed this feature. In Unity/Flexbox4Unity, **only GameObjects that have a FlexItem component will ever be resized**.

(note: if a GameObject only has a FlexContainer component, **it will not be resized**; FlexContainers control how their children are resized, they never resize themselves)

Percentages are applied to parent width OR height

For historical reasons, CSS applies any “PERCENT” relative to the **width** (even if it’s a height!). Except … some parts of CSS don’t, and apply height to height, width to width!

This is far too confusing, and it makes layout much more difficult when designing GUIs, so I ignored the CSS specification. In Flexbox4Unity, **percent-widths apply to the width** and **percent-heights apply to the height**, always.

Not supported yet: flex-wrap

If you want this feature, please contact me via support – currently, no-one is using flex-wrap in actual games, and we need some people with real apps/games to beta-test it and help find any bugs.

Using Flexbox for rich, complex, GUIs

Online guides

If you've not used Flexbox before in CSS/HTML, I strongly recommend you read this simple guide to flexbox (contains lots of useful images and simple explanations):

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Note: the source-code examples in that page are not necessary – everything is handled for you in the Unity Editor GUI. But the images and text explain how you can use flexbox to position things.

Quick overview

With flexbox, you group things together in ***FlexContainers***, and use multiple nested containers to achieve any layout you want.

Each container draws its children either in a ***row***, or in a ***column***.

By default, the container is set to 'grow=1', which means that any remaining space will be used to 'grow' the child items, and set to 'align=stretch', which means that all items in a row will be max height, or all items in a column will be max width.

For finer control, each child has a ***FlexItem*** attached that lets you fine-tune details such as padding or margins around that child (not available in LITE version).

Help!

If you get stuck, try the official Unity Forum:

https://forum.unity.com/threads/released-flexbox-fast-easy-layout-from-html-css-in-unity-2017-2018-2019.699749/

Troubleshooting

… “It's not working! What do I do?!”

FAQ: Support forum

https://forum.unity.com/threads/released-flexbox-fast-easy-layout-from-html-css-in-unity-2017-2018-2019.699749/

FAQ: My GUI scales weirdly when I change resolution

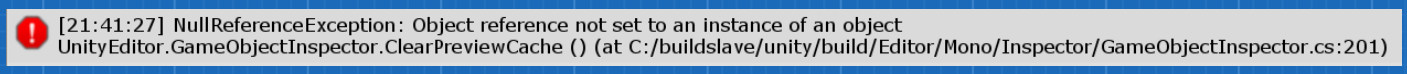
Unity version 5.x upwards requires that when you instantiate Prefabs and parent them, you use this code:

```
GameObject go = // anything here, e.g.: Instantiate( prefab );
go.transform.SetParent( parentTransform, false ); // this line is critical
```

Unity has some old bugs in their SetParent code that are only resolved using the extra, optional, flag “false”. They cannot fix them because old games depend on the old behaviour, so they recommend all new games use the new behaviours by sending “false” as a second parameter.

FAQ: “NullReferenceException: Object reference not set to an instance of an object”

If you see this every time you re-compile your project’s C# files:



...please report it to Unity directly!

There is currently nothing I can do about this, it is a bug inside the UnityEditor (if you look closely, you’ll see the class with the error is one of Unity’s own internal classes)

Recent changes

2.4: Improvements to VR support, new Experimental flex-wrap

- FIXED: player builds weren't exporting settings, were freezing layout on build
- Added auto-integration of UnityUI.RawImage

2.3: Complete re-write of the Layout engine (fixed many bugs + edge-cases)

2.2: Improved editing GUI

- Greatly improved the Inspector window for FlexItem and FlexContainer
- Added Help tab and "Advanced" inspector for direct access to all flex params

2.1: New layout-algorithms

- Adopted CSS3's names for classes and variables: FlexItem and FlexContainer
- Created modular, user-replaceable, Layout algorithm system

2.0: Big improvements for Unity 2018, Unity 2019

- Fixed: auto-refresh stopped working due to changes internally in Unity
- Added: Live-preview system to visualise complex layouts in Scene View
- Added: integrated Unity's new ProjectSettings system for storing and editing settings
- Added: free registration system for faster customer-support

1.5: Replaced UnityUI's slow algorithm with a custom fast one

- Improved: integration with Unity 2018.x editor versions

1.4: Added "padding" and improved auto-sized content

1.3: Performance optimizations

1.2: Greatly improved "margins" support

1.1: Fix Unity prefabs error, improve Aspect-Ratio Fill

1.0: First public release

First release!