Contents

3. Introduction

This manual is made primary for figma designers, but is detailed enough for anyone who had never used Figma before. In the future, the manual will be updated and refined.

You can send your comments and suggestions to the official telegram of D.A. Assets - https://t.me/da_assets, as well as get help from members of our community and from me personally to the official group of this asset in the telegram - https://t.me/figma_unity_converter.

STARTING WITH **VERSION 1.0.7** default separator is "**-**", but you can use "**/**" separator if your layout is already adapted to the old separator.
Also, number of **changes** have been made regarding the **tag-system**.
Now, in **most** cases, you **don't need** use **img** tag, and you **never** need to **use** the following tags:
frame, cont, vert, hor, fimg.

I **recommend** that you **read** this **manual** again.

4. Naming and tags

In order for the component to be **imported correctly**, you need to add a **tag** to the component **name**, in the format “**tag / name / info**” or “**tag - name - info**”.

Most components don't need a tag.

Actual tag list:

- img
- btn
- bg
- field
- pholder

1. “**tag**” - tag of the component. It is required to determine the type of component when imported into a Unity project.

2. “-” or “/” - required separators. Separates the tag from everything after it.
Only one type of **separator** can be **used** within your **layout**.

3. “**name**” - component name. Specify the logical purpose of the component here, for example "**btn - menu open**".

4. “**info**” - any information you want to include in the name of this component, or a random set of characters in order to **avoid duplicate names** for components that are **not the same**. If two different components have the **same name** - when importing, one will be **replaced** with another, and the layout will **deteriorate**.

5. Examples with “**-**” separator:

btn - menu open - black silver
btn - menu open - white red
btn - menu open - blue yellow
bg - circle pattern - used in frame 1, 2, 3
img - avatar - main user
img - avatar - user 1
img - avatar - user 2

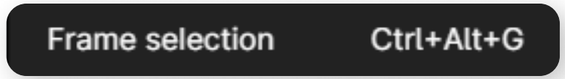
Examples with “**/**” separator:

btn / menu open / black silver
btn / menu open / white red
btn / menu open / blue yellow
bg / circle pattern / used in frame 1, 2, 3
img / avatar / main user
img / avatar / user 1
img / avatar / user 2

5. Layout rules

1. **Identical** elements have the **same** name, **different** ones are **different**

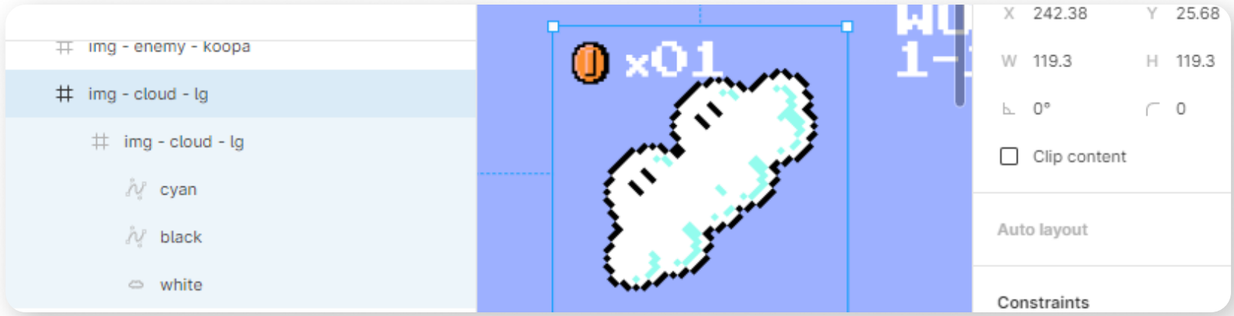
2. A **container** component that **contains** other components **must** be created using “**Frame section**”**.**



3. Make sure that the "**Rotation**" for components that have a tag is equal to **0** (zero), **otherwise** the layout will **not be imported** correctly. Components that make up a tagged component can have any rotation angle. For example, if you need to create a component **rotated** by a certain **degree**, set the "**Rotation**" property to the **degree** you need, then turn this component **into** a **frame**, and **assign** a **tag** to this frame, for example, "**img**" if the component is an image in the context of the UI.

Not correct:

Correct:



5.1. Layout rules

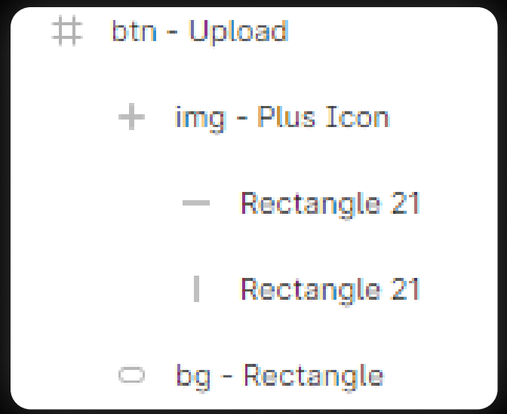
4. The text after the tag **should not** consist only of **special characters.**

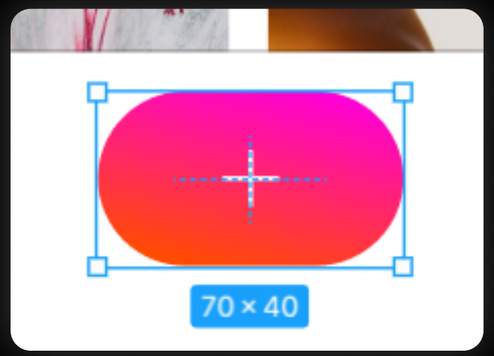
   Not correct: **btn - +**
   Correct: **btn - plus**

5. If a component is self-sufficient, it is a **single component** in its essence (it can be a separate whole **image** or an **icon**, or a **background**), it can be marked with a special tag ("**img**" or "**bg**", you can read more about these tags in corresponding sections), and **include components** with any other **tags**, in this case, these internal components **will not** be perceived by the importer as **separate**, and **will be part** of a single **icon** or **image**.
   For more information, **see** section “**Image**”.

6. Background

bg / White Background





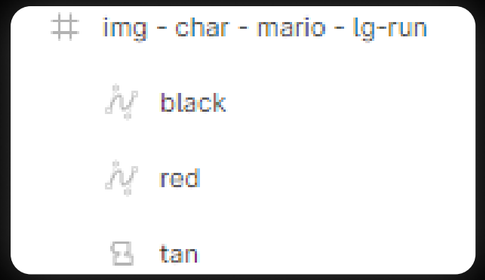
1. The component with the "**bg**" tag can be part of the components with the "**frame**" and "**button**" tags. Constraints of the component with the "**bg**" tag must be set as "**Left and Right** & **Top and Bottom**"

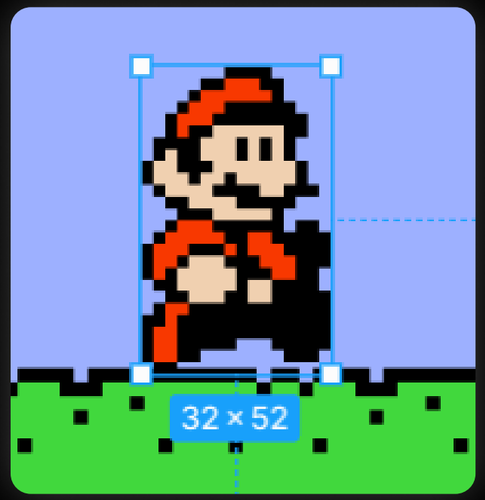
2. Example naming:

bg - pink bg
bg - for level frames
bg - for red buttons

7. Image

img / char / mario / lg-run



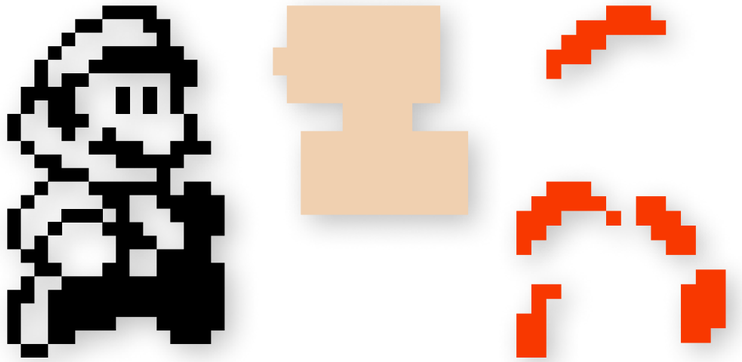


1. The "**img**" tag applies to any **single image** that is a part of the layout.
It can contain other images, vectors and other elements: regardless of the content of the component with the tag "**img**", it will be perceived as a **single** component.

2. This is how the exported component will look like:

With img tag:

Without img tag:

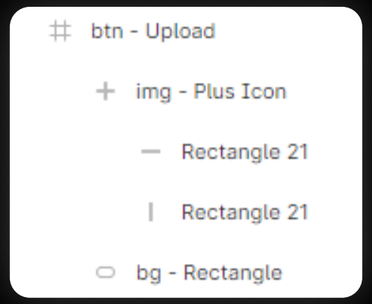


8. Button and BetterButton

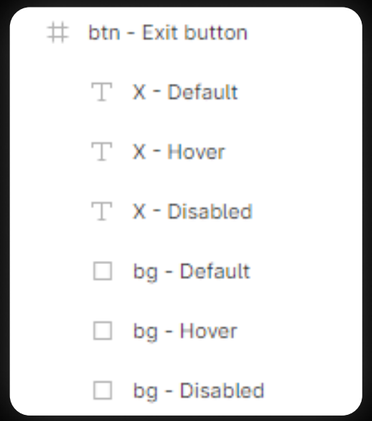
btn / Upload

A button can be created using two different structures.

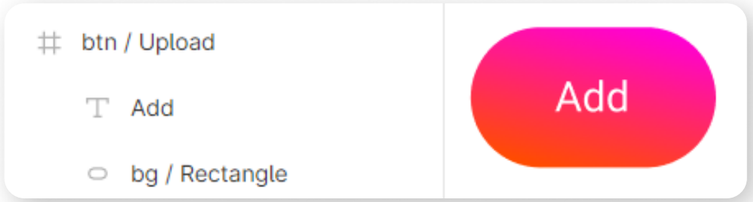
1

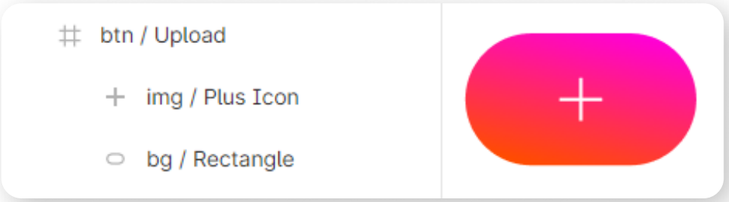


2



1 The button has to be imported as a **standard** component of the Unity **button**. It can consist of a **background** with a "**bg**" tag, and **text** (without a tag) or an **image** (with a "**img**" tag).



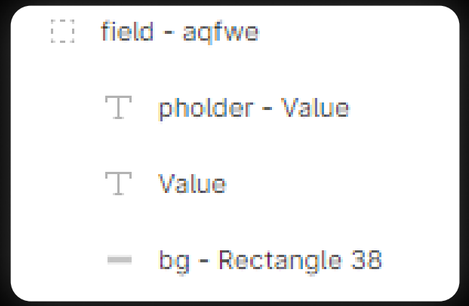


1 The button that will be imported into Unity using the "**BetterButton**" component. By default, this component is in the mode of changing the image when you click on it, or disabling this component in Unity (**Sprite Swap** mode). For such a component to be imported correctly, create options inside it for various states, namely: **Default**, **Hover** and **Disabled**. For each of the listed states, a text component (or an image tagged with "**img**") must be created, and a background component with a tag "**bg**".

9. Input Field

field / Login

pholder / Placeholder

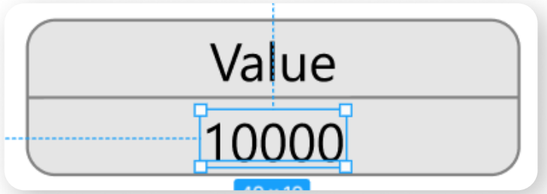


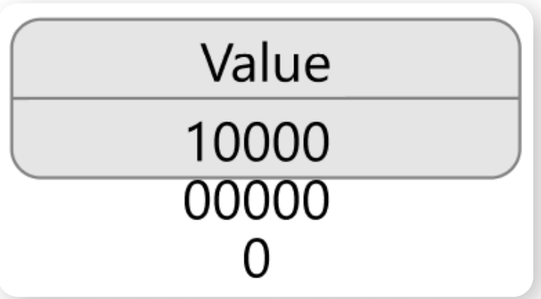
1. The "**field**" tag is used to create a **Input Field** component in Unity. It consists of a **background**, a **text** component (**without** a tag) that displays the **entered text**, and a **text** component with a "**pholder**" tag - a **placeholder** (tag "pholder" outside Input Field is **not** used).

10. Working with text fields

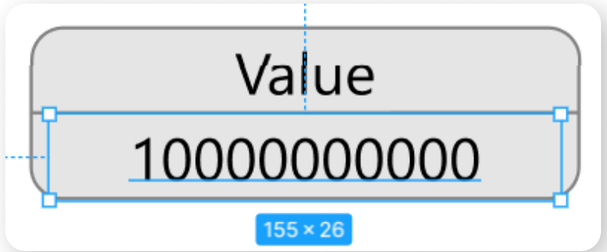
When you create a field for input, display of text, or value, then take into account the size of the text selection frame. It must be large enough so that when the amount of text increases or decreases, it remains within its bounds.

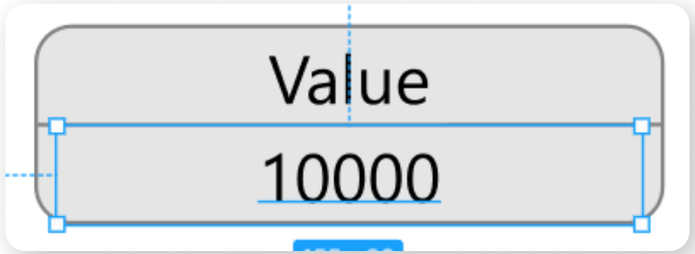
1. Below is an example of how not to do it. If in the future, inside Unity, using an algorithm or manually, we want to set this text component to the value "1000000000", it will not fit and will get out of bounds.





1. Below is an example of the correct size for a text component.

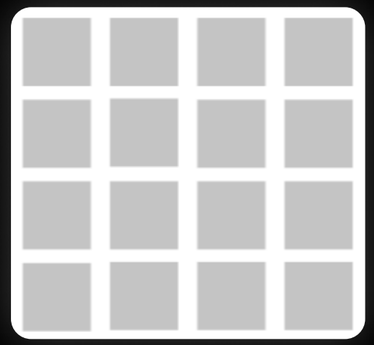


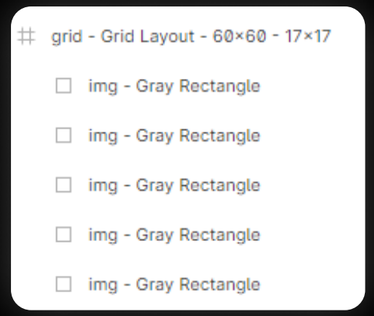


11. Grid Layout

grid / Grid Layout / 60x60 / 17x17

1 2 3 4



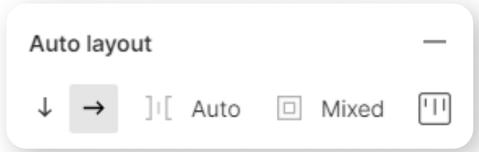


1. Tag "grid" is used to create component **Grid Layout Group** in Unity.
2. Enter the name or purpose of the component here.
3. Specify the cell size in **WIDTH**x**HEIGHT** format.
4. Specify the spacing between cells in **X**x**Y** format.

12. Horizontal Layout

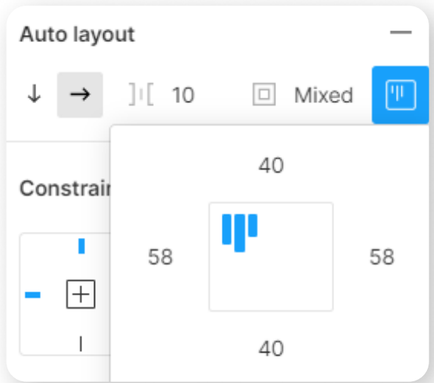
1. To create this component, activate the property "**Auto Layout**".

2. Switch the component to **Horizontal** mode.



3. Set horizontal **padding**.

4. Set indents.

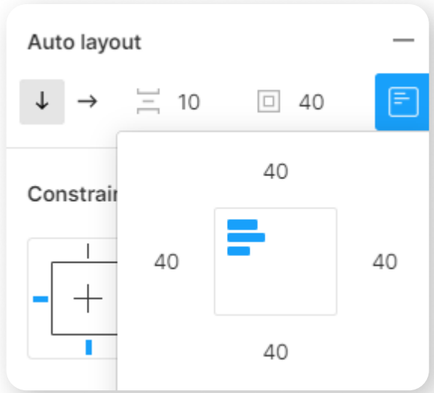


13. Vertical Layout

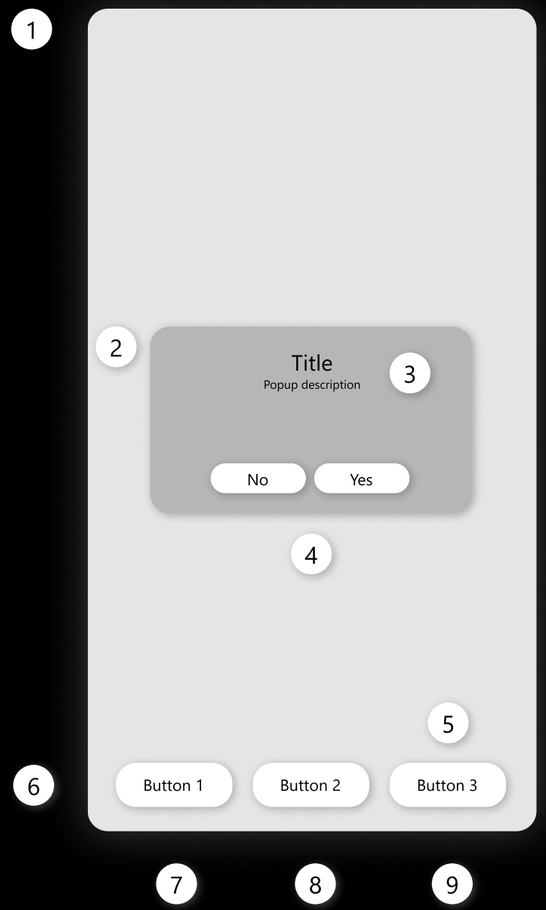
1. To create this component, activate the property "**Auto Layout**".

3. Set vertical **padding**.

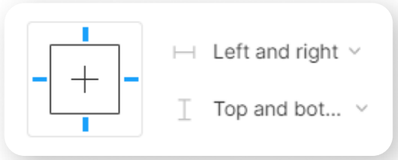
4. Set indents.



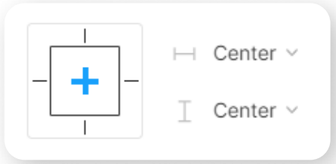
14. Constraints



1 **Constraints** for the **background** of the frame.

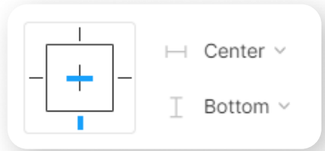


2

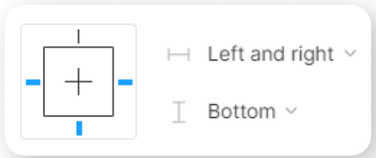


3

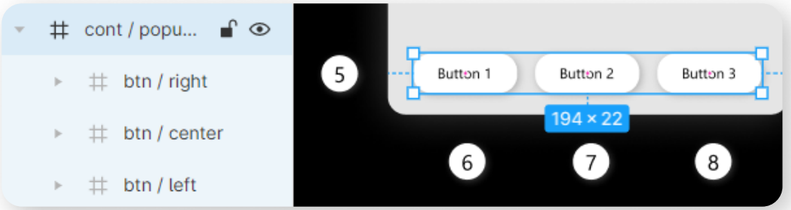
4

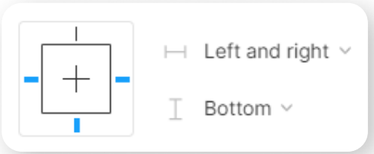


5 **Constraints** for the **text** compontent.



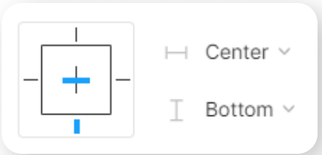
6



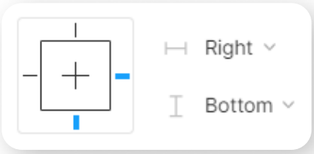


7

8



9

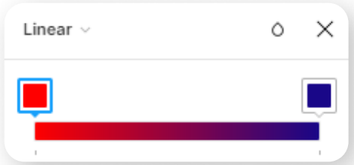


D.A. Assets

15. Modern Procedural UI Kit and Procedural UI Image

1. If you are importing a project using these assets, then you can use the corner radius.

2. Also, in **MPUIKit** you can use a **linear gradient**. All **other** types of gradients are **not supported**.
Transparency in gradients **is not supported** by the **MPUIKit** asset.



3. **Transparency** in gradients **is not supported** by the **MPUIKit** asset.

4. **Procedural UI Image** only supports simple **fill** with transparency, and **does not support** any kind of **gradient**.

15.1. Modern Procedural UI Kit

3 With these assets, the **image** you create in Figma with a gradient and fill is **recreated** with this asset **inside Unity**. For this reason, it is not required to store the image file in the folder with the Unity project, this allows the weight of your application.
However, **not all images** using a gradient or fill can be **recreated**.
Complex shaped assets **cannot** be **recreated**, and **vector icons** too - they will be replaced with white quads.
To prevent this from happening, they **need** to be marked with **img** tag.