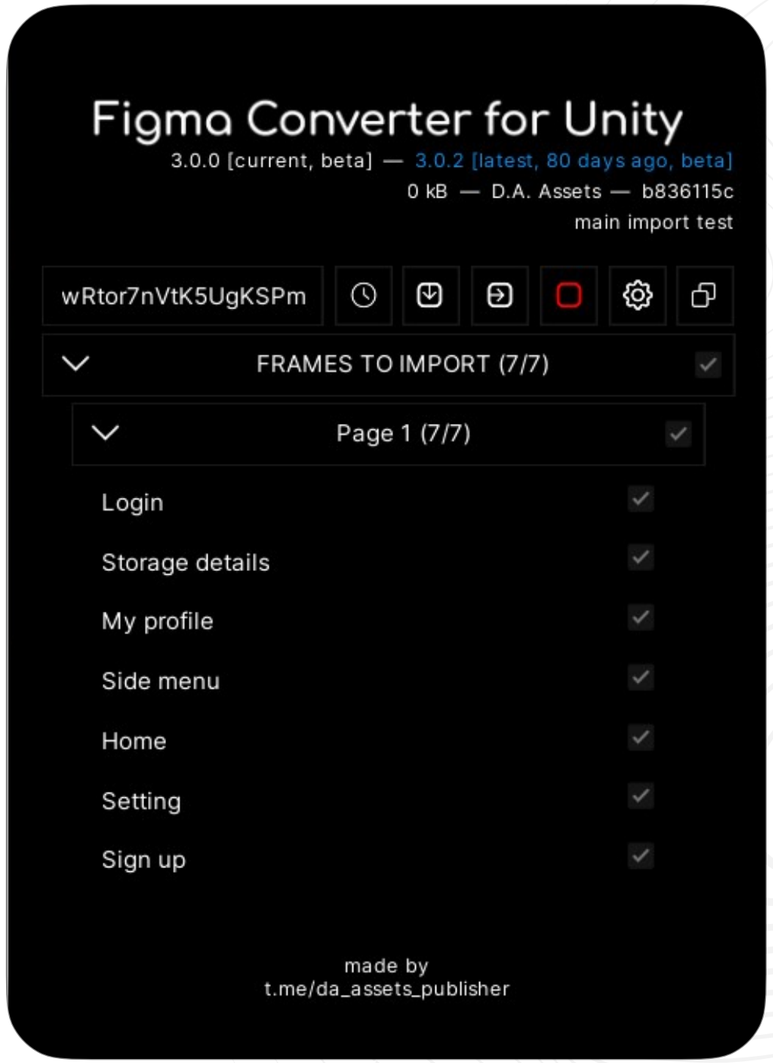


Figma Converter for Unity

manual for designers

3.1.5

Contents

Introduction

I recommend reading this manual
in its entirety.

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.

If you see any errors in the manual, or oddities or errors in the operation of the asset, please report it to me using known contacts.
As a rule, I quickly respond to messages and provide assistance, as well as fix bugs in the next update.
You can leave comments about the features that you want to see in the asset - it's will also be considered.

Telegram Support: https://t.me/da_assets
Email Support: da.assets.publisher@gmail.com
Telegram Group: https://t.me/figma_unity_converter
Discord Server: https://discord.com/invite/ZsnDffV5eE
Website: https://da-assets.github.io/site/

Change Log

2.0.0 You need to **re-read** the entire **manual**.
Now you can use either of the two delimiters "**-**" or "**/**" in your layout. When importing, the delimiter "**/**" will be replaced with "**-**".
Added "**cont**" tag.
Removed "**bg**" tag.

2.0.5 Updated the "**Layout rules**" section in the current manual.

2.1.0 Different components with the same name now do not overwrite each other on import.
You can give components the same name.
See "**Uniqueness of the components**" slide for details.

3.0.7 Removed "**grid**" tag.

3.0.9 **FCU Button** is replaced with **D.A. Button**, naming instructions in **D.A. Button** section have been updated.

3.1.2 "pholder" tag has been replaced with "ph" tag.
"field" tag has been replaced with "fld" tag.

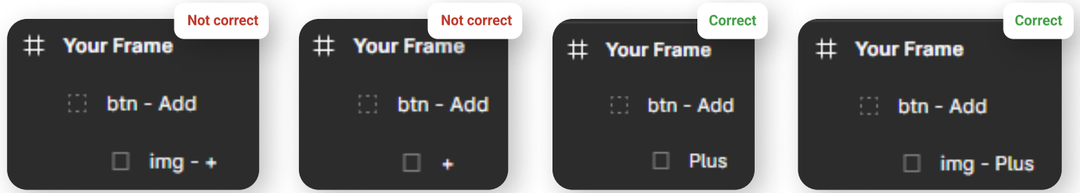
3.1.3 Added information about **Sections** import to the "**Layout rules**" section.

3.1.4 Actual tag list for UITK has been added.

3.1.5 Updated buttons section.

Layout rules

1. Disable "**Clip Content**" on frames before importing.
2. To enable the asset to import **Sections**, apply "**Frame selection**" to them (place the section inside a frame).
3. The text after the tag **should not** consist only of **special characters.**



4. If you wanted a certain component to be imported as a single image, and this did not happen by default, see the **Image** slide.
5. If you want the components within your component to not be merged into a single image, see the **Container** slide.
6. In some cases, your layout may look **incorrect** after importing - this is **because** you are **placing fills and effects on** components that are the **parent** of other separately imported components.

Two ways to solve the problem:

1. Manually changing the layout, **making** the parent's **fill** and **effects** as a separate component that is **inside** the **parent**.
2. Using **third-party assets** which can **procedurally recreate** your components.

You need to use third party assets if:

1. Not all components of your layout were imported.
2. You don't want to change your layout so that all components of your layout are imported.

Layout rules

It is important to consider:

• If all the properties of your Figma component in the table are marked as "**Yes**", the component will be procedurally recreated on the scene, i.e. will not be downloaded.

• If your component has properties with at least one "**No**", it will be downloaded as an image.

• If your component has at least one "**No**" and is a fill on a parent component that contains other separately imported components, Unity will not add an image component to it.

Asset	Rectangle	Circle	Solid Fill	Gradient	Other Fills	Stroke	Corner Radius	Layer Blur	Other Effects
Shapes2D	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No
Modern Procedural UI Kit	Yes	Yes	Yes	No	No	Yes	Yes	No	No
Procedural UI Image	Yes	Yes	Yes	No	No	No	Yes	No	No
Unity Image	Yes	No	Yes	No	No	No	No	No	No

Example 1: You have a Rectangle with Fill.

If your asset is "**Unity Image**":
1. The component will not be downloaded
2. The component will be recreated
3. A component can be on a parent component

Example 2: You have a Rectangle component with a fill and rounded corners.

If your asset is "**Unity Image**":
1. The component will not be downloaded
2. The component will not be recreated
3. Component cannot be on parent component

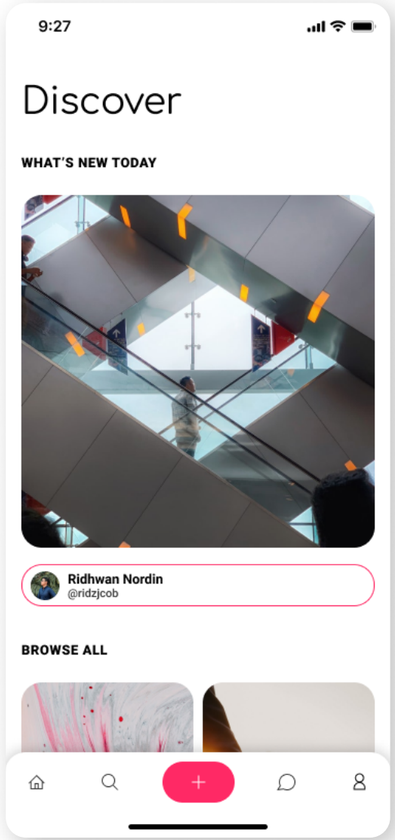
Layout rules

If your asset is "**Shapes2D**":
1. The component will not be downloaded
2. The component will be recreated
3. A component can be on a parent component

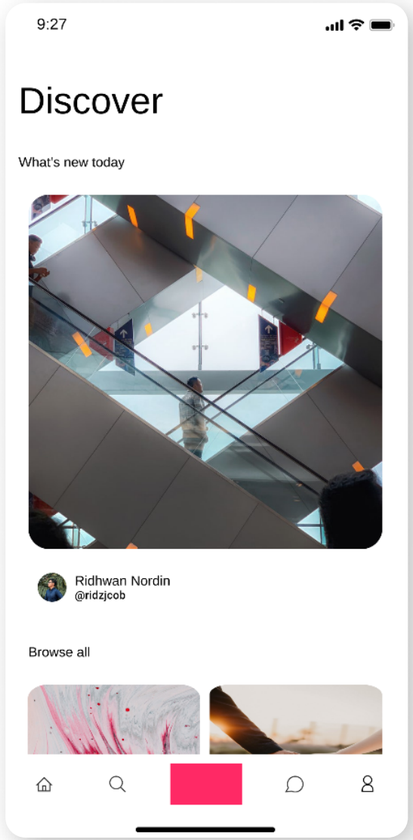
Example 3: importing a frame from a project with examples.

An example of an import with the "**Unity Image**":

Frame in Figma:



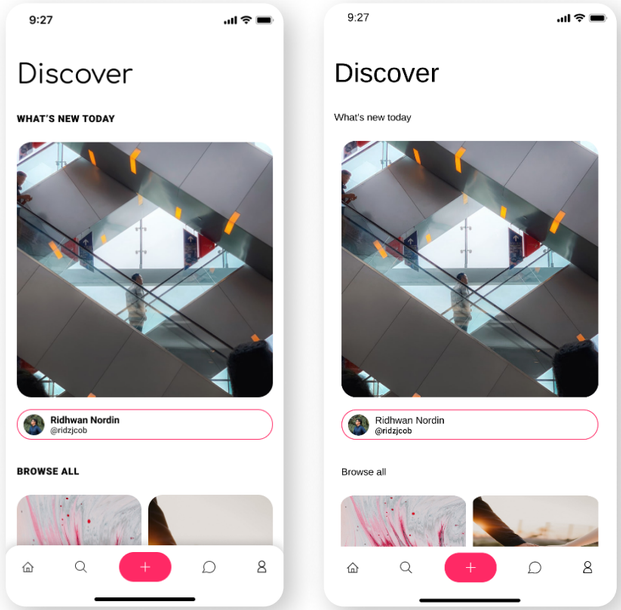
Frame in Unity:



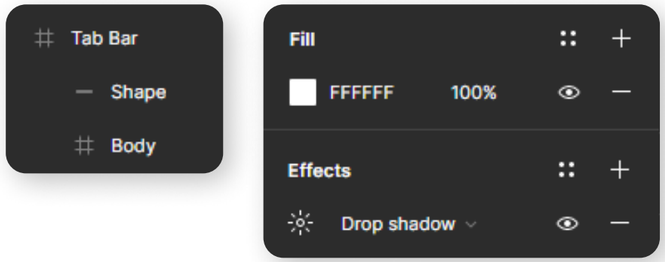
Layout rules

An example of an import with the "**Shapes2D**" asset:

Frame in Figma: **Frame in Unity with "Shape":**

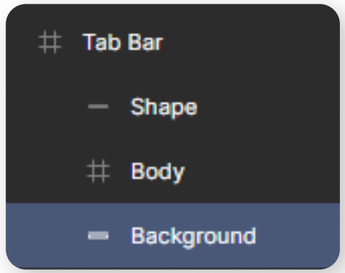


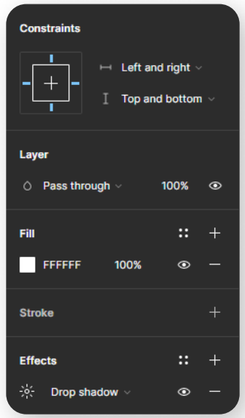
As you can see, there are still differences - there is no shadow at the bottom menu.
Let's see why this happened. Let's move on to this component in Figma:



Layout rules

It turns out that the container for other objects - "**Tab bar**", contains the shadow effect. If you try to export it as an image to keep the shadow, you end up with an image that includes both the shadow and all of its children. We do not need this, because the child components are buttons, and we want to import them separately.
To solve this situation, you can use the "**TrueShadow**" asset (you can read more about this in the section "**Shadows and True Shadow asset**" in the manual for designers) - then the shadow component will be automatically recreated using this asset and you won't need to make changes to the layout. If you don't want to use "**TrueShadow**" asset, follow the instructions below:
Remove the background and shadow of "**Top bar**" component, and implement them as a separate component.
Let's create this component and name it "**Background**", then add a fill and a shadow to it.

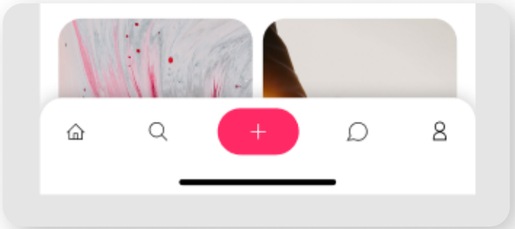




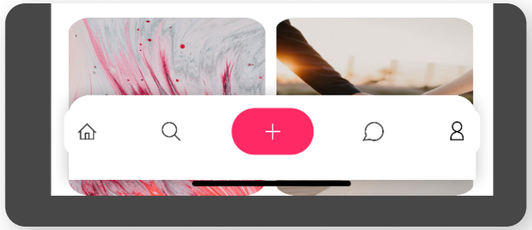
Layout rules

Let's try to import the frame again:

Frame in Figma:

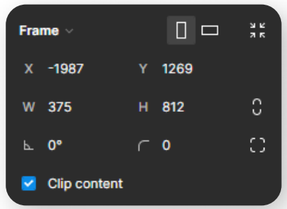


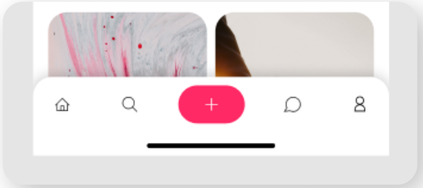
Frame in Unity with "Shape":



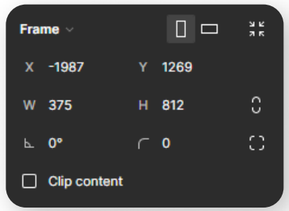
Something went wrong again. The object with the shadow was imported separately, but after importing it, it became smaller than in Figma.
This is due to the "**Clip Content**" feature in Figma. Let's turn it off for our frame.

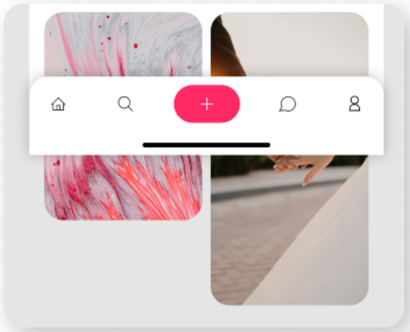
Before disabling "Clip Content":





After disabling "Clip Content":

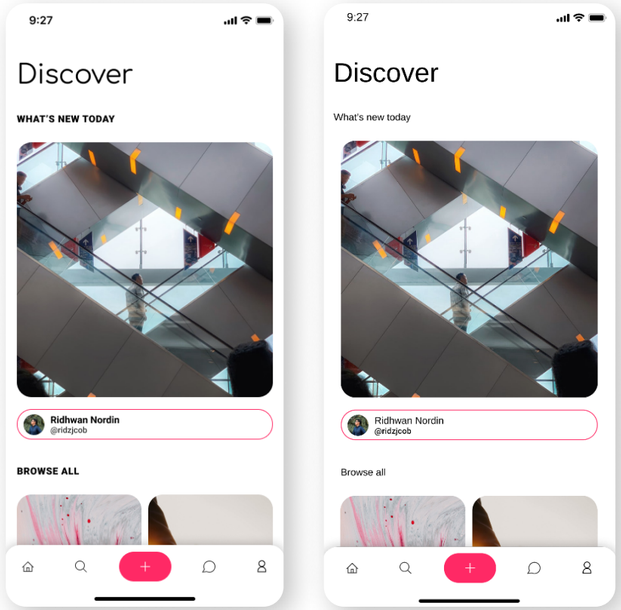




Layout rules

Let's try to import the frame again:

Frame in Figma: **Frame in Unity with "Shape":**



Now perfect!

As you can see, sometimes you need to make small changes to bring the import to the desired quality.

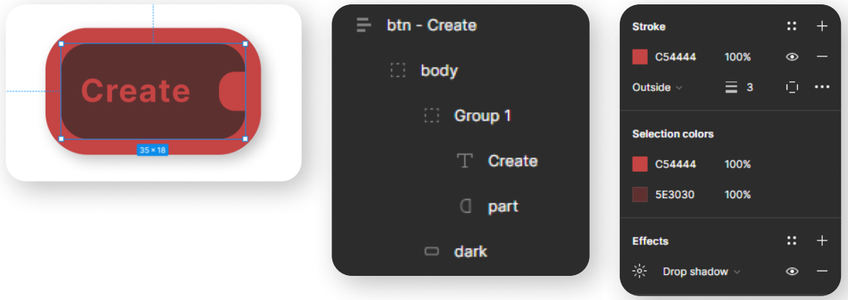
If you have any questions, write to me.

Layout rules

Example 4: incorrect and correct structure.

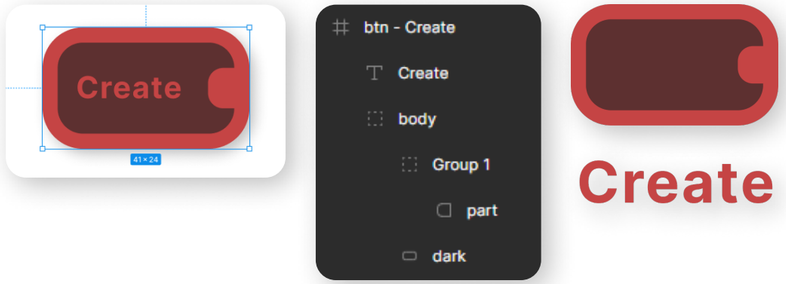
An **example** of an **incorrect structure**. What is done wrong?

- The stroke extends beyond the boundaries of the parent object.
- The stroke is assigned to the parent component.
- The shadow is assigned to the parent component.
- The text is inside the background.



An **example** of a **correct structure**.

- The parent component has no fills or effects assigned.
- The stroke does not extend beyond the boundaries of the parent object.
- The stroke is assigned to the body component, which is the background of the button.
- The shadow is assigned to the body component.



Naming and tags

Actual tag list for Canvas:

- img
- cont
- btn
- fld
- ph

Actual tag list for UITK:

- img
- cont
- btn

D.A. Assets

Tags are needed to **clarify the import** of your layout, **or** to **assign scripts** to some objects in Unity.
Component name format with tag: "**tag - name - info**".
Most components don't need a tag.

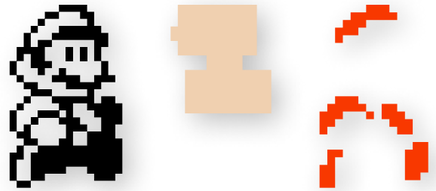
1. "**tag**" - tag of the component. It is needed so that the importer can understand exactly how to import this object. The **tag is part of** the component **name**.
2. "-" or "/" - required separators. Separates the tag from everything after it.
3. "**name**" - component name. Specify the logical purpose of the component here, for example "**btn - menu open**".
4. "**info**" - any information you want to include in the name of this component.
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
6. If tags have been renamed in the new version of the asset and you are experiencing compatibility issues, you can ask the developers to change the values in the "**Figma Tag**" field in the "**Tags**" array in the "**FcuConfig**" file.

Container

1. If your object is imported as a single image, but you want its internal structure to be imported, give it a "**cont**" tag.

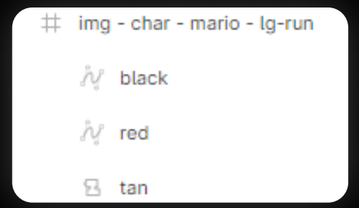
With cont tag:

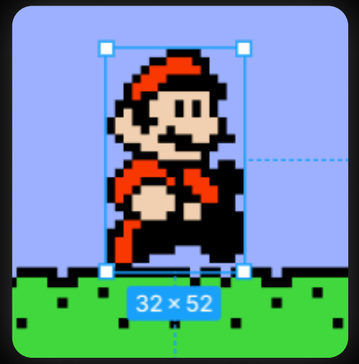


Without cont tag:

Image

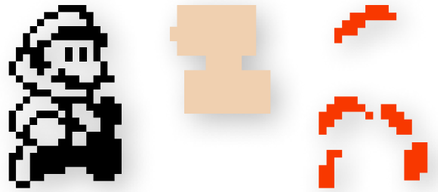
img - char - mario - lg-run





1. Apply this tag to a parent object if you want that component, along with its child components, to become a single image when imported.

Without img tag:

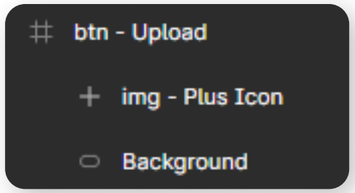


With img tag:

2. If all children of the parent component are vectors, the parent component automatically will import as a single image.

Button
FcuButton

1. In order for the button script to be added to it's GameObject during import, you need to add the "**btn**" tag to the beginning of its name in Figma.

2. 

   If you use the "**Unity Button**" without states (def, pres etc.), and you want the "**Color Tint**" animation to be applied to the button (changing color when pressed), the button must be assigned a "**Target Graphic**" - usually an image that is the background of the button.
   In order for the "**Target Graphic**" to be assigned automatically during import, the **image** that is the **background** of the button **must be at the end of the hierarchy** of its child objects.

2. **For each** of supported button assets, you can specify special behavior for states "**pressed**", "**hovered**", "**disabled**" (and "**selected**" for ”Unity Button” and ”FcuButton”).
   For this, you need to assign special tags to the button's child components.

   Special tags:
   def - default button state, used when nothing happens to the button;
   pres - used when a button is clicked;
   hov - used when hovering over a button;
   dis - used when a button is disabled (when it cannot be clicked).

3. Please take a look at the scheme showing how objects with tags are assigned to the button component.

Button
FcuButton

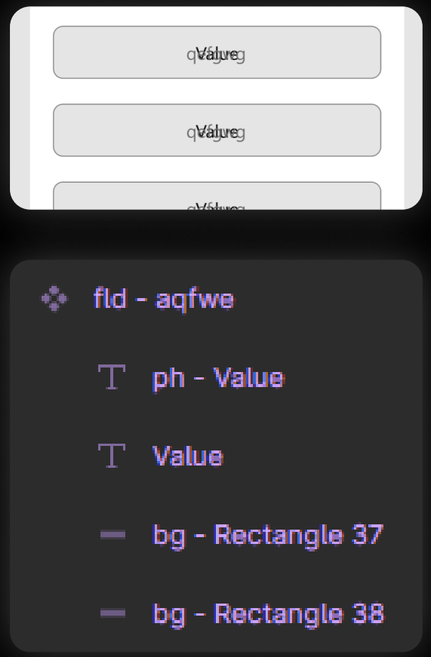
4 As you can see, the component with the "**def**" tag is recognized as TargetGraphic, and from the other components with state tags, only the color is taken, which will then be assigned to the component with the "**def**" tag in Unity.

5 When the button is in **ColorTint** mode, **child objects** of the button with tags "hov", "pres", "sel", "dis" are removed from the scene.

6 If your **TargetGraphic** component cannot be recreated on the scene using Unity's means or with procedural assets, it will be downloaded, and the button transition type will be switched to **SpriteSwap** mode.
In **SpriteSwap** mode, child **image objects** of the button are not removed, regardless of the tag.

Input Field

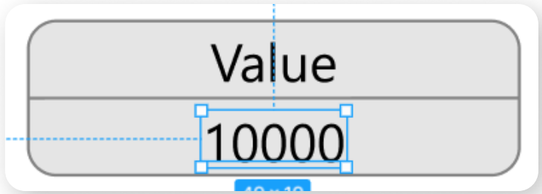
fld - Username

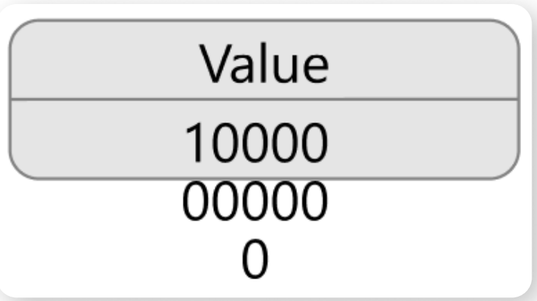


1. The "**fld**" tag is used to create a "**Input Field**" component in Unity. It consists of a **background**, a **text** component (**without** a tag) that displays the **entered text**, and a **text** component with a "**ph**" tag - a **placeholder** (tag "ph" outside Input Field is **not** used).

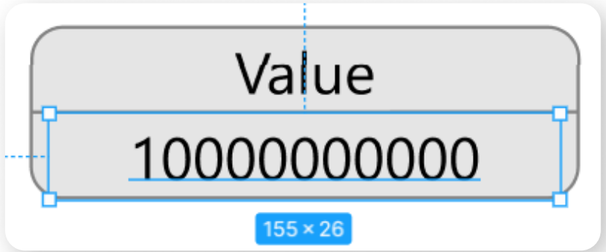
Text

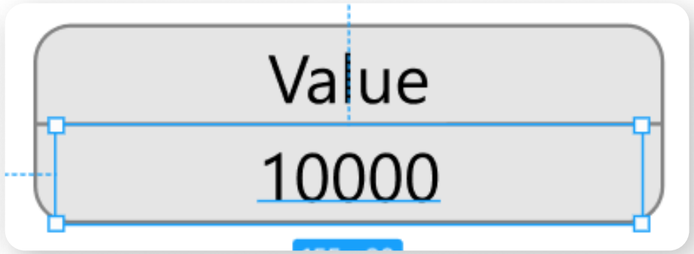
1. Example of how not to do it. If in the future, inside Unity, using an algorithm or manually, we want to set this text component to the value "1000000000", it will not fit and will get out of bounds.





2. Example of the correct size for a text component.

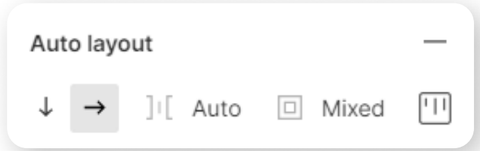




Horizontal Layout

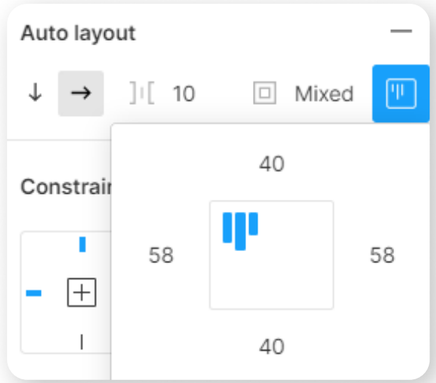
1. To create this component, activate the property "**Auto Layout**".

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



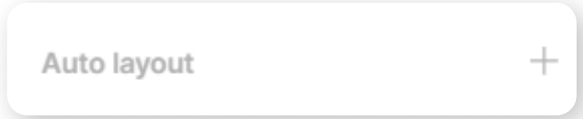
3. Set horizontal **padding**.

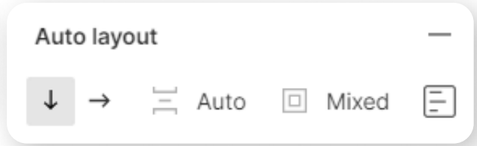
4. Set indents.



Vertical Layout

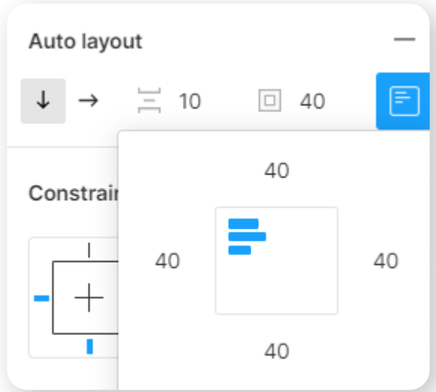
1. To create this component, activate the property "**Auto Layout**".





2. Set vertical **padding**.

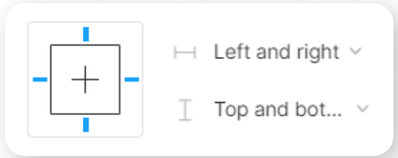
3. Set indents.



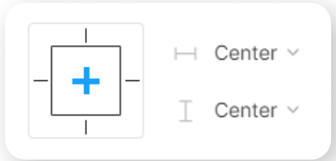
Constraints

You can set **Constraints** for your layout elements at the layout scene, and then it's will be **automatically transferred to Unity.**
Let's take a look at **constraints** assigned to a frame from a project with examples (**Example templates.fig**).

1. Frame **background** set to "**Left and right**" & "**Top and bottom**", to make the background stretch to match the size of the frame.



2. The popup in the center of the frame is set to "**Center**" & "**Center**" so that it stays in the center of the screen when the frame is resized.

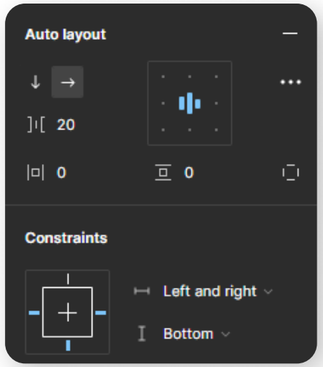


3. "**Left and right**" & "**Top and bottom**" is assigned to the container containing the title and text of the popup, so that when the popup is resized, it will stretch to the size of the popup.
"**Left and right**" & "**Top**" is assigned to the header so that when the popup is resized, it stays at the top of the popup and has a fixed height.
"**Left and right**" & "**Top and bottom**" is assigned to the text of the popup, so that when the size of the popup is changed, it will grow and contain more text.

4. "**Left and right**" & "**Bottom**" is assigned to the button container so that it stretches across the width of the popup and stays at the bottom of the popup.
"**Scale**" & "**Top and bottom**" is assigned to the popup buttons so that the buttons stretch based on the width of the popup.

Constraints

5 "**Left and right**" & "**Bottom**" is assigned to the bottom menu button container so that it spans the width of the frame and sits at the bottom of the frame.
The container is assigned an "**AutoLayout**" with **center alignment** so that the buttons always remain in the center of the container, and, accordingly, in the center of the frame along the X axis.



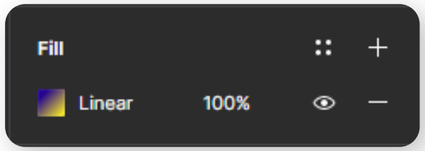
Shapes2D, MPUIKit, Procedural UI Image

With these assets, the image you create in Figma with a simple 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.

List of images that cannot be recreated (will be downloaded):

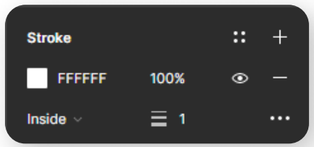
1. Complex images that have an internal structure.

2. Images that don’t use a solid fill.



3. Images that are not a circle or a rectangle.

4. If you are using asset "**Procedural UI Image**" - stroked images.

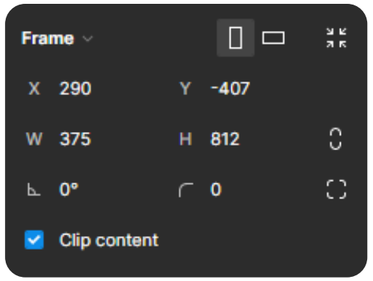


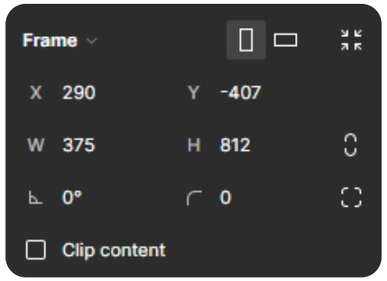
D.A. Assets

Shadows and True Shadow asset

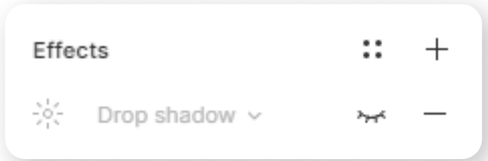
Video manual:
https://www.youtube.com/watch?v=ckyS96RmsY8

1. Starting with version 2.0.0, the **asset supports importing Figma shadows**, so the **shadow** component **don't need to be disabled** before the layout is imported.
If the Figma's shadow extends beyond the container containing the shadow component, the component will be distorted.
To prevent this, **disable** "**Clip content**" feature **for** your **frames**.





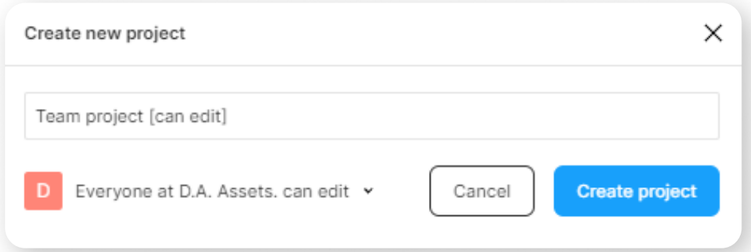
2. If you want to use "**TrueShadow**" asset, you **need to disable** the component's shadow before importing the layout, otherwise "**TrueShadow**" asset's shadow will duplicate the Figma's shadow, and overlay it.



D.A. Assets

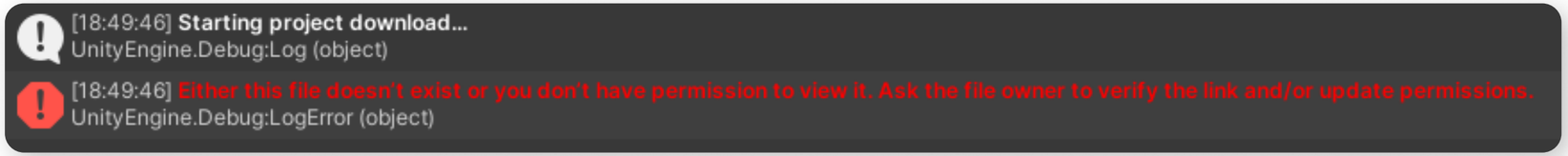
Teamwork

1. You can create a team project with the following **permissions**:
   - can edit
   - can view
   - invite-only



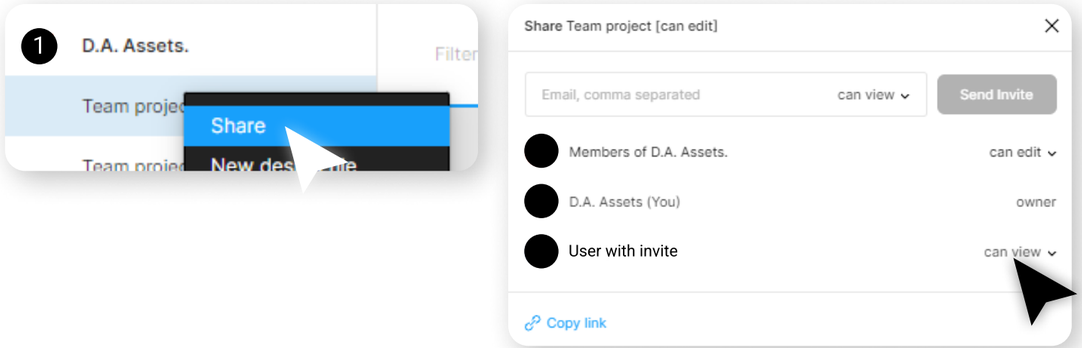
Regardless of which permission you choose, if you select the **account** from which this **project** was **created** when **authorizing** inside an **asset** in Unity, you will be able to **import frames** from it.
But in order for the frames to be imported through **other accounts** that you want to **give access to the project**, you need to follow certain steps. See below.

2. Consider a team-project with the "**can edit**" permission.
If user **A** is the **owner** of the project, and user **B** tries to download the project from the link, and at the same time, user **B** does **not have acces**s to the project, an error will occur during import:

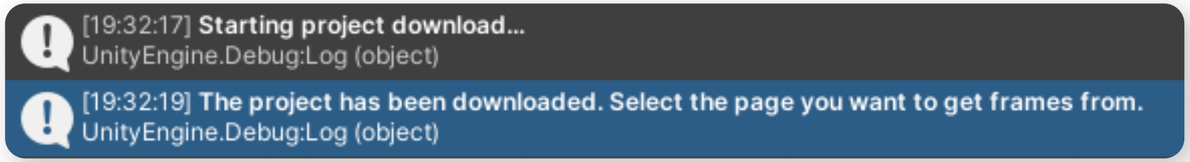


Teamwork

3 Send an invite to the **email** of the **user** you want to **give access** to the project.
In order for this user to be able to import a project using an asset, the "**can view**" permission is enough.



4 **After** the user **accepts** an **invite** with the correct permissions, he will be able to **import** the project.



5 If, for some reason, you have an **error** with permissions, or layout pictures are not downloading, **try re-inviting** the user who imports the layout.
Also, some users were helped by **re-creating** the project with the components of the **design system** and the project in which **instances** of these **components** are **used**.
I have kept the original wording of the "**Either this file doesn't exist**" error, so information on it **can be found** on the **Figma forum** or on Google. You may be able to find more precise instructions on how to resolve the permission problem for your situation.
If none of the above helped and the project does not import - create your own project using the account you are logged into the asset, copy the necessary frames to this project, and try to import this project.

Uniqueness of the components

To prevent components with the same names from **overwriting** each other **after downloading**, a random set of numbers is added to their names before downloading, which is calculated based on the properties of these components and all the components that are inside it.
If these properties change, the set of numbers also changes, and then a new file will be downloaded.

To avoid duplicating prefabs and images in your project, use the component creation function in Figma.

Since version 2.1.0 you **can give** components the **same name**, and if the properties of these components and their child components are different, it’s will **not overwrite each other.**